Transport Layer Optimizations for Heterogeneous Wireless Multimedia Networks

A Thesis
Presented to
The Academic Faculty

by

Antonios Argyriou

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

School of Electrical and Computer Engineering
Georgia Institute of Technology
December 2005

Transport Layer Optimizations for Heterogeneous Wireless Multimedia Networks

Approved by:

Professor Vijay Madisetti, Advisor
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Yucel Altunbasak
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Raghupathy Sivakumar
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Sudhakar Yalamanchili
School of Electrical and Computer Engineering
Georgia Institute of Technology

Professor Umakishore Ramachandran
College of Computing
Georgia Institute of Technology

Date Approved: August 19, 2005

To my family.

ACKNOWLEDGEMENTS

I would like to thank my advisor Dr. Madisetti, for his assistance, continuing support, and encouragement throughout my years in the Ph.D. program. I would also like to thank Dr. Sivakumar, and Dr. Altunbasak, for serving in my reading committee and for their suggestions during the preparation of this thesis.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

CBR	Constant Bit Rate.
FEC	Forward Error Correction.
GPRS	General Packet Radio Service.
IETF	Internet Engineering Task Force.
MAN	Metropolitan Area Network.
MPEG	Motion Picture Experts Group.
MSS	Maximum Segment Size.
OFDM	Orthogonal Frequency-Division Multiplexing.
QoS	Quality of Service.
RAN	Radio Access Networks.
RTP	Real-Time Transport Protocol.
SCTP	Stream Control Transmission Protocol.
TCP	Transmission Control Protocol.
TFRC	TCP Friendly Rate Control.
UMTS	Universal Mobile Telecommunications System.
WLAN	Wireless Local Area Network.

SUMMARY

The explosive growth of the Internet during the last few years, has been propelled by the TCP/IP protocol suite and the best effort packet forwarding service. However, quality of service (QoS) is far from being a reality especially for multimedia services like video streaming and video conferencing. In the case of wireless and mobile networks, the problem becomes even worse due to the physics of the medium, resulting into further deterioration of the system performance.

Goal of this dissertation is the systematic development of comprehensive models that jointly characterize the performance of transport protocols and media delivery in heterogeneous wireless networks. At the core of our novel methodology, is the use of analytical models for driving the design of media transport algorithms, so that the delivery of conversational and non-interactive multimedia data is enhanced in terms of throughput, delay, and jitter. More specifically, we develop analytical models that characterize the throughput and goodput of the transmission control protocol (TCP) and the transmission friendly rate control (TFRC) protocol, when CBR and VBR multimedia workloads are considered. Subsequently, we enhance the transport protocol models with new parameters that capture the playback buffer performance and the expected video distortion at the receiver. In this way a complete end-to-end model for media streaming is obtained. This model is used as a basis for a new algorithm for rate-distortion optimized mode selection in video streaming applications. As a next step, we extend the developed models for the aforementioned protocols, so that heterogeneous wireless networks can be accommodated. Subsequently, new algorithms are proposed in order to enhance the developed media streaming algorithms when heterogeneous wireless networks are also included. Finally, the aforementioned models and algorithms are extended for the case of concurrent multipath media transport over several hybrid wired/wireless links.

CHAPTER I

INTRODUCTION

During the past few years, we have witnessed an unprecedented growth in the number of wireless users, applications, and network access technologies. Typical mobile users nowadays, have access to various types of wireless networks that may be cellular networks, wireless local area networks (WLAN), metropolitan area networks (MAN), home networks, or even mobile ad-hoc networks (figure 1). These networks are usually accessible through different types of terminals (mobile phones, PDAs, notebooks). However, as the wireless and mobile market matures from the early adopters to normal users, new services will be demanded. These demands are converging towards the demands that exist for wired telecommunications services. With the availability of these wireless services, the location is no longer of importance to private users. Thus, the demand of mobile users connected over wireless networks will approach this mixture of services. With the omnipresence of wireless services, the usage schemes will become independent from location and connection type.

In this mixed environment, feature-rich real-time media applications like video conferencing, videophony, and video streaming will have to be supported. These kinds of applications have strict end-to-end delay constraints, which is usually less than 200 milliseconds. However, as wireless networks are quickly becoming an important component of the modern communications infrastructure, IP will be the key technology that will drive the unification of the next generation mobile/wireless systems by being able to support high-speed data, Internet access, and multimedia streaming on all-IP networks. Therefore, CDMA2000 and UMTS 3G cellular networks [1], metropolitan area networks like 802.16 [103], and high bit-rate WLANs (54 and 11 Mbps in IEEE 802.11a and 802.11b respectively) will coexist in the future. All the above-mentioned cutting-edge developments however, confront the high technical hurdles associated with high bit rate, quality of service (QoS), and real-time requirements of video applications [110, 104]. Therefore, the traditional protocol stacks

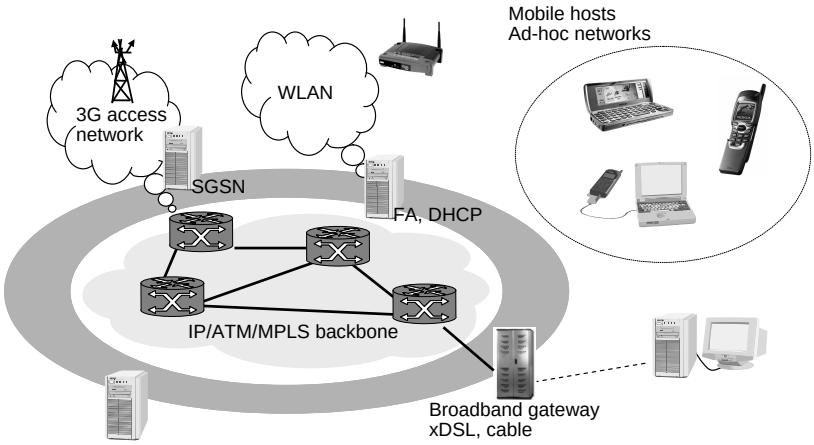


Figure 1: Heterogeneous wireless network architecture.

have to be re-engineered, by designing more flexible and generic communication protocols.

In this dissertation we are concerned with protocols that operate at the transport layer, as defined as part of the OSI stack [78], and their interactions with media applications. More specifically, we focus on TCP, and two recently standardized IETF protocols, namely the stream control transmission protocol (SCTP) [101], and the TCP-friendly rate control protocol (TFRC) [46]. Our objective is to design algorithms that optimize wireless multimedia delivery by carefully considering the behavior of the aforementioned transport protocols. Initially we develop stochastic models that characterize the performance of these transport protocols for a variety of wireless inter-networking scenarios. Subsequently, by following the end-to-end principle [23], we present the design of algorithms that push all the necessary functionality to the endpoints. Our protocols handle video streaming functions at the endpoints in an end-to-end fashion and optimize crucial metrics such as latency, bandwidth utilization, and jitter, between two hosts that communicate with a unicast session over wired/wireless IP-based networks. An outline of every chapter in this dissertation is given next, while figure 2 visualizes the organization of this dissertation.

Transport Protocol Models for CBR and VBR Workloads: In this part of the

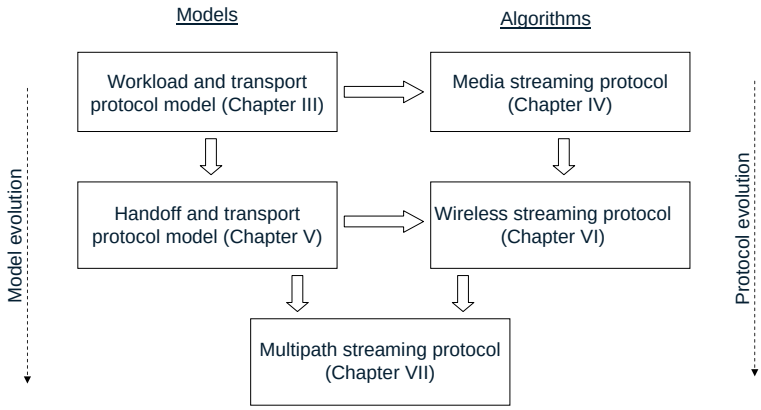


Figure 2: Dissertation organization.

dissertation, we develop analytical models that characterize TCP and TFRC throughput for different traffic workloads, namely CBR, VBR and bulk traffic. These traffic types, are representative of typical multimedia workloads. Our work is the first one that considers the effect of these types of workloads in TCP and TFRC throughput models. We compare the developed models with existing ones, and we demonstrate that the assumption of flows with an infinite data backlog, may significantly affect the TCP throughput estimate in case of CBR and VBR workloads. Subsequently, we analytically derive packet delivery bounds for TCP under the aforementioned workloads, and we present experimental results that highlight TCP's performance. The analytical derivation of packet delivery bounds for TCP and TFRC is crucial, since it unveils the ability of the protocol's algorithms to deliver a specific level of QoS for a given set of network conditions. Finally, with the help of the developed models and experimental results, we identify certain cases where TFRC is not able to support CBR and VBR workloads effectively. We demonstrate that with the proposed model, these predictions can be more accurate, leading to a better understanding of the protocol and workload interactions.

Rate-Distortion Optimized Unicast Video Streaming with TCP: In the next

chapter of this dissertation, we initially present an analytical study that characterizes the performance of video streaming with the transmission control protocol (TCP). First, we develop an analytical model of the expected video distortion at the decoder with respect to the TCP parameters, channel state, and error concealment method at the receiver. Based on this model, we propose an algorithm for RD optimized mode selection (RDOMS) for video streaming with TCP. We conduct extensive experimental results and we demonstrate that for real-time video streaming, we can observe PSNR improvement in the range of 2 db over currently proposed TCP-based streaming mechanisms. Our next contribution is the development of a joint model of the TCP protocol, and the playback buffer at the receiver. This model is developed by utilizing parts of the model that we developed in chapter 3. Based on this new model, we derive the optimal playback rate at the decoder. Subsequently, based on the two models, we propose an algorithm, for RD optimized packet scheduling with TCP. Our results show an additional improvement of nearly one db, when packet scheduling is applied together with the RDOMS algorithm.

Modeling the Effect of Handoffs on Transport Protocol Performance: After we have completed a joint performance model and protocol for wireline networks, in this chapter we develop from scratch a comprehensive model that captures transport protocol performance in mobile networks. More specifically, we are concerned the effects of mobility on the performance of transport protocols. Therefore, we develop analytical models that characterize the throughput, latency, and jitter of TCP and TFRC protocols as a function of the handoff induced packet loss rate and the disruption time. The result of our analysis is a modular performance evaluation model, that may be used for analyzing the effect of various mobility management architectures and mobility scenarios for existing and emerging transport protocols. We also introduce the concept of the "recovery period", which is defined as the time required for the transport protocol to achieve the nominal throughput in a new link, after a handoff. This parameter allows even more precise analysis of the effects of handoffs on the steady state throughput of TCP and TFRC, and consequently to the media application. Ultimately, the precise modeling of the effect of handoffs, will help on the design of optimized multimedia streaming policies.

Video Streaming in Heterogeneous Mobile Wireless Networks: This chapter of the dissertation brings together the wireline models and algorithms developed in chapter 4, with the mobility performance model developed in the previous chapter. We formalize the process of video streaming in heterogeneous wireless networks, by emphasizing on the performance model of the transport protocol in use. Our comprehensive model led to the development of a new protocol for end-to-end video streaming in a heterogeneous wireless environment. In the second part of this chapter we propose a new proactive media handoff protocol, which is combined with the previously developed streaming protocol so that it can assure the best possible QoS for a media streaming session. The proposed protocol, is implemented on top of SCTP, and employs several mechanisms that use end-to-end semantics for signaling handoffs, transmitting control messages, and traffic redirection. With this work we demonstrate that the use of analytical, closed-form performance models of transport protocols, can be utilized by a practical video streaming protocol.

Multipath Transport Protocol Models for Wireless Video Streaming: In this final chapter, we are concerned with the use of TCP for multipath video streaming. Our objective is to demonstrate that the use of analytical performance models can be used for driving the behavior of a multipath video streaming protocol, so that the delivered video quality is improved. To achieve this objective, we initially develop a stochastic closed-form latency model, that captures the behavior of TCP when multipath transport is considered. Based on the previously developed models, we propose three algorithms for optimizing video streaming. More specifically, we initially present an adaptive playback adaption algorithm that operates only at the client without intervention of the sender. The second algorithm controls multipath scheduling of video packets, and can operate on top of any multipath transport protocol. Main task of this algorithm, is the estimation of expected latencies of video packets, and the proper allocation to the outgoing paths, based on the playback deadlines at the client. Finally, we introduce the idea of multipath retransmission, and a new algorithm, that intelligently decides the allocation of video packet retransmissions to the available paths.

CHAPTER II

BACKGROUND

In this chapter we introduce the basic technologies that are in the focus of this dissertation. Initially we provide an overview of the main mobile/wireless network technologies today. Next, we analyze the operation of TCP, and two recently IETF standardized transport protocols, namely the stream control transmission protocol (SCTP) [101], and the TCP-friendly rate control protocol (TFRC) [46]. We will describe their main characteristics, and outline their behavior in wireless mobile network environments. After the description of the transport protocols, we proceed with the analysis of the main components of multimedia communication systems, and we highlight their interactions with transport protocols. More specifically, we will describe two popular media delivery mechanisms, namely pre-recorded and real-time video streaming. These are the two application classes, that we attempt to jointly optimize in this dissertation, under the aforementioned transport protocols.

2.1 Mobile and Wireless Networks

An understanding of the existing and emerging mobile network technologies is crucial so that the proper behavior of transport protocols can be designed. We will briefly describe in this section, the main mobile and wireless network technologies that exist today. Details concerning these networks, will be given in subsequent chapters of this dissertation.

The main difference between wired and wireless access networks, is that wired links provide one-to-one communication without interference, whereas wireless links use one-to-many communication that suffer from noise, significant interference, and bandwidth limitations. These problems are caused by the time-varying and frequency selective nature of the wireless channels. These channel fluctuations are the result of a combination of attenuation (free space propagation), multipath fading and shadowing [86]. In this rather harsh environment, several wireless access networks architectures have been defined [103]. Main feature of both

existing and emerging wireless radio access networks (RAN), is the increased set of differences in the specifications of the physical (PHY) and medium access control (MAC) layers. This trend stems from the advantages that different PHY/MAC layers offer according to the target application domain. It is technically impossible to capture the needs of several application scenarios with a small set of RANs. Therefore it is a expected that these technologies will coexist for the near future and will act complementary to each other. For example this heterogeneity of RANs, is now believed to be essential for the next generation mobile networks (fourth generation (4G)) which will operate on Internet standardized technologies combined with various access technologies such as WLAN/3G/MAN. This heterogeneous network, will be able to provide speeds ranging from 100 Mb/s in cellular networks to 1 Gb/s in hot-spot networks [103]. In order to ensure connection ubiquity together with high bandwidth and mobility, the network architecture must be heterogeneous rather than homogeneous. 4G technologies include management of handovers (within the same RAN technology, and across different RAN technologies) and, thus, involve alternation of network quality of service (QoS) (e.g., bandwidth, delay).

In table 1 we can see the main characteristics at layers 1 and 2 of existing RANs. The first cellular technologies devised for packet data services after the success of the 2G technology were GPRS and EDGE that are also known as 2.5G. The newest 3G cellular technologies, use CDMA for the air interface, since this solution provides better performance for voice traffic. However, the 3GPP and 3GPP2 standardization organizations, have developed solutions for supporting data traffic. For WCDMA (UMTS) the high speed downlink packet access (HSDPA) [48, 59] technology has been developed, while for CDMA2000 a similar solution is called 1xEV-DO [14]. In HSDPA the channel bandwidth allocation between voice and data can be configured, and a single carrier mechanism that shares codes and transmission power is also a possible configuration. For 1xEV-DO however, a channel must be exclusively allocated to data traffic, but the use of smaller bandwidth channels overcomes this limitation. Recently, the 1xEV-DV technology for CDMA2000, uses a dynamic algorithm for allocating the channel between voice and data traffic [97].

For data oriented networks however, like WLANs, high bit-rates are usually achieved

by the use of the orthogonal frequency-division multiplexing (OFDM) technology as the air interface. The most important high speed data-oriented WLAN standards like 802.11a/g and Hiperlan [21, 103], o metropolitan arean networks like 802.16a [50] use this technology. OFDM divides a single high bit-rate channel into several narrowband channels that transmit in parallel. The rationale behind the use of multiple narrowband channels, is that this solution minimizes the effects of multipath delay spread. So OFDM is inherently better for high-peak-rate packet data. While an OFDM proposal was made for the 3G air interface [48], its was rejected due to the large peak-to-average power ratio of the OFDM signal, leading thus to increased power consumption by the mobile.

The precise technical details of all the access network technologies are beyond the scope of this dissertation.

Table 1: Comparison of radio access networks.

Radio access	PHY (Kbps)	Channel bandwidth	Modulation	Channel coding	Error recovery
WCDMA	64-384	5 MHz	QPSK/BPSK	Conv./Turbo	srARQ
CDMA2000	1.2-307	1/5 MHz	QPSK/16QAM	Conv./Turbo	srARQ
GPRS	9-171	200 KHz	GMSK	Convolutional	srARQ
EDGE	8.8-473.6	200 MHz	GMSK/8PSK	Convolutional	hARQ
802.11	6-54 x10^3	20-22 MHz	QPSK/BPSK	Convolutional	swARQ

The heterogeneity that these mobile networks introduce, and their affect on media delivery/transport protocols, will concern us in this dissertation. We will analyze both the problems that introduce for the transport of media, but we will also exploit the possible benefits that this heterogeneity introduces.

2.2 Internet Transport Protocols

We have already stressed the importance of transport protocols as part of a communications stack. In this section, we will describe their behavior that will unveil their importance as part of any communications stack, including media communications systems. The understanding of the protocol behavior, will allow us to rethink new approaches for the design of end-to-end media communications in wireless IP-based networks.

2.2.1 Transmission Control Protocol

Initial purpose of the transmission control protocol (TCP), was to provide reliable datagram delivery over wireline connectionless packet-based networks. However, as the technology evolved, the new mobile and wireless world has set new challenges that have to be met by TCP. In this subsection we will outline TCP's functionality, and we will briefly describe its behavior in mobile environments.

TCP was initially defined in RFC 793 [84], and provides a connection-oriented and reliable byte stream oriented transport service. The term connection-oriented means that whenever applications want to transfer data, they establish a logical connection between two endpoints. This explicit TCP connection is established by a three-way handshake process [84]. When data is passed from an application to TCP for delivery, TCP splits the data stream into smaller chunks, and adds a protocol information header to form a segment. The largest chunk of data that TCP can include in each segment is limited by the maximum segment size (MSS). During the initialization of a connection, each host advertises its MSS, and TCP chooses the smallest value to avoid further fragmentation. These segments are passed to IP, and in turn IP appends its own header information to form datagrams or packets.

Another interesting feature of TCP, is the way its probing for network resources. TCP is using a window-based congestion avoidance mechanism, that acts as a self-clocking regulator based on feedback from the receiver. When the TCP receiver successfully receives a packet, it sends an acknowledgment (ACK) back to the sender. Main task of the sender, is to keep a current record of the number of unacknowledged packets that it has released into the network, which is called the congestion window. In addition, the sender keeps an estimate of the round-trip time (RTT). The TCP sender is increasing its window size as long as packets are being acknowledged. The way the sender detects a packet loss is by either the non-arrival of a packet ACK within a certain time (i.e. via timer expiry or time out), or by the arrival of multiple ACKs with the same next expected packet number (typically 3 duplicate ACKs). We will distinguish these two modes of packet loss detection by using the names "TO" and "TD" respectively. With TCP, a packet loss is interpreted by the sender

always as an indication of congestion, and this results into reduction of the window size, thereby indirectly controlling the data rate. Every time TCP receives an ACK, it updates its estimate of the RTT. We assume that the RTT estimate is constant. Hence, in normal operation, the timer is set to this value each time a packet is transmitted. However, when multiple TO events take place consecutively (i.e. without the reception of any ACKs in between), TCP applies the exponential backoff algorithm, where, for the k-th consecutive TO event (k is an integer > 0), the packet is retransmitted and the timer is set to the minimum of the values 2^k RTO and 64RTO.

The behavior of TCP in wireless mobile scenarios, has attracted considerable research, which we will analyze in detail in subsequent chapters of this dissertation. In general however, TCP performance in mobile/wireless networks suffers from a series of problems. One of the main research results identify TCP's inability to distinguish between wireless and congestion induced packet losses [103]. Another problem is related to blackouts, due to disconnections, that lead to exponential increase of the RTO [99, 10]. A mechanism for partly resolving this problem is through explicit layer 2 notifications to TCP, so that it can freeze the RTO [47, 75]. An import concern is also the long and fluctuating delays due to local retransmissions in the wireless links [26, 28], or due to deep buffering in cellular access networks [28]. This situation can result into the invocation of the congestion control algorithm and substantial decrease in the throughput. Finally, performance is deteriorated due fragmentation caused by the smaller packet sizes that usually characterize wireless networks [103].

In this dissertation we will investigate the performance of TCP in wireless scenarios, not however for the general case of data transport, but in the context of media applications. We will explore the use of TCP for media delivery, and we will propose cross-layer enhancements between TCP and the media streaming application.

2.2.2 TCP Friendly Rate Control

The significant interest around the widely used TCP, has generated a wealth of research that has produced several models that capture its behavior [5, 80, 95]. Utilizing the analyses

by one of the closed form models, a new congestion-control protocol has emerged and is called TCP-friendly rate control (TFRC) [46]. TFRC is not a full-fledged transport protocol. However, it controls the transmission rate of non-TCP traffic and it sets it at a rate similar to the rate that TCP would send data if the TCP flow were experiencing the same mean round-trip time and packet loss probability. It has been shown that the TFRC protocol achieves TCP-friendliness while it prevents unnecessary bandwidth fluctuations, by estimation of the packet loss rate and consequently the allowed output rate. TFRC is using a closed form equation for TCP throughput in order to regulate the sender's output rate as a function of the packet loss rate p [46]:

$$T = \frac{s}{RTT\sqrt{\frac{2p}{3}} + RTO_0(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{1}$$

In this equation s is the packet size, RTT is the RTT estimate, and RTO_0 is the value of the retransmission timer. In addition TFRC follows a packet spacing algorithm at the sender:

$$t_{inter} = \frac{s\sqrt{RTT_{cur}}}{T * M} \tag{2}$$

where M is the average of the square roots of the RTTs calculated using an explicit window moving average (EWMA), and RTT_{cur} is the most recent RTT sample [46]. Equation 1 does not represent the actual TFRC sending rate, but only an upper bound for it. The actual output rate of TFRC may is calculated using a different algorithm which we will describe in a later part of this dissertation. If no packet loss has yet been "seen" by the sender, TFRC emulates the slow start algorithm of TCP by doubling the transmission rate every RTT. In addition, the TFRC algorithm, assures that the output rate is not doubled more than once during an RTT, similar to TCP [99]. The average receive rate at the receiver, is also used as part of the TFRC rate estimation algorithm.

Overall, TFRC is emerging as an important protocol for the delivery of real-time media data in the Internet. However, its performance and the relative merits of its various design approaches have not been properly addressed in the case of heterogeneous mobile networks.

In this dissertation we will investigate this behavior, by developing first a comprehensive model, and subsequently analyze its capability to deliver real-time media data.

2.2.3 Stream Control Transmission Protocol

The final protocol that we will describe, is the stream control transmission protocol (SCTP) and it was also developed by IETF. SCTP is reliable, connection-oriented transport protocol that was initially designed for SS7 signaling transport [101]. However, it soon become obvious that it has general applicability as a transport protocol that can operate on top of connectionless packet networks such as the Internet similar to TCP and UDP. The first interesting characteristic of SCTP is the packet format, in which the payload is not transported as a unified chunk of data as in TCP, but in the form of well defined and self-contained messages that are called chunks. There are several types of chunks. A user message is formed into data chunks which have their own set of flags and length. Several control chunks exist and they can be inserted into the same SCTP packet with data chunks. The rationale behind this protocol format, is that head-of-line blocking of unrelated user messages is avoided. Another advantage is that SCTP is able to decouple reliable delivery from message ordering by introducing the idea of streams. A stream, is an abstraction that allows applications to preserve in-order reliable delivery within a stream, but unordered delivery across streams. In this way, head-of-line blocking is avoided at the receiver in case multiple independent data streams are flowing in the same SCTP session. This is different form the chunk-based mechanism, since these streams carry user data which are transported in data chunks.

Connection establishment in SCTP is quite different from that of its counterpart TCP and it requires four steps before it is completed. An INIT message is sent from the active opener while the receiver replies with an INIT_ACK which contains a message authentication code [101]. Subsequently, the sender sends a COOKIE-ECHO chunk where it echoes back to the passive opener, the cookie that received. When the passive opener receives this cookie back it checks its validity, and then the host actually allocates the resources and the association is established. This form of connection establishment prevents denial of service

attacks with spoofed IP addresses [101].

Another new feature that SCTP introduced, was that of explicit support for multi-homed hosts. This means that a single SCTP session can use alternatively anyone of the available IP addresses of a host without disrupting an ongoing session. This feature is currently used by SCTP only as a backup mechanism that helps to recover from link failures. SCTP can identify these failures because it maintains a state for each remote IP address by sending heartbeat messages periodically. Despite all these differences from TCP, the congestion control algorithm is a window-based AIMD as in TCP, primarily for achieving TCP friendliness [101, 99, 52]. One minor difference is located in the fast retransmit algorithm which needs now four duplicate acknowledgments before it retransmits the presumably lost packet.

A recent extension to the SCTP protocol called partial reliability for SCTP (PR-SCTP [100]), is of particular use for the applications we are concerned with. The authors describe an extension to the SCTP that allows an endpoint to inform its peer that it should move the cumulative ack point (CumTSNack) forward [1]. In case both endpoints of the SCTP association support this extension, it can be used by an implementation to provide partially reliable data transmission service to an upper layer protocol. The authors present the protocol extensions which can be summed up in a new parameter for the initial session setup messages (INIT and INIT_ACK messages), and the definition of a new FORWARD_TSN message type, that provides explicit control over the receiver's CumTSNack.

We envision SCTP as a successor to TCP for applications that have rather complicated data transport requirements. We will use the multihoming feature of SCTP, as a flexible protocol platform that allows more efficient implementation of the algorithms developed for TCP and TFRC.

2.3 Multimedia Communication Systems

In this section we will describe the main components of media communications systems. Understanding of media applications, is necessary so that the relative tradeoffs are properly

[1] Cumulative TSN Ack Point: Equivalent to TCP's largest sequence number received.

identified when a cross-layer optimized system is designed (figure 3).

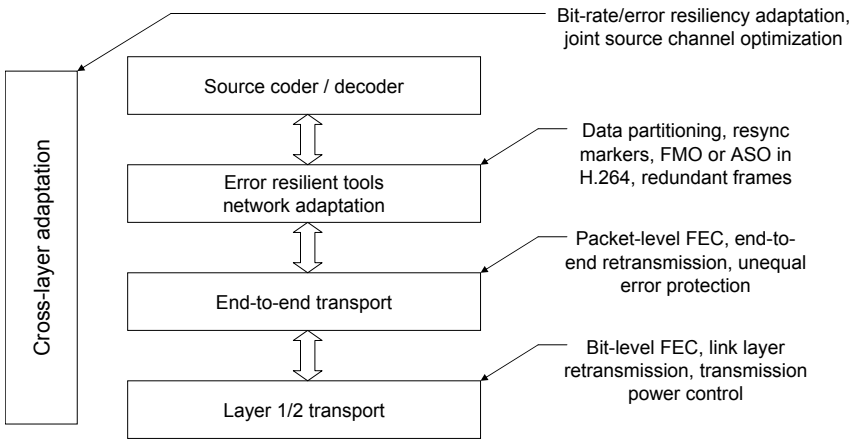


Figure 3: Cross layer design for multimedia communication systems [37].

A typical video communication system has five components (figure 4): First is the source encoder that compresses video and audio signals into media packets. These packets can be sent to lower system layers, or stored for transmission on demand. Second is the application layer which is in charge of channel coding and the packetization functions. Third is the transport layer that performs congestion control and delivers media packets from the sender to the receiver, while it assures fair network resource utilization. Fourth is the delivery of media packets to the client through the transport protocol. And the final fifth component is the the decompression and display of the video units at the receiver.

The typical flow of events in a media session is described next: At the sender, the video encoder generates video packets. The source bit rate is constrained by a rate controller which makes the bitrate allocation either at the frame or packet level. The bit rate constraint is usually set based on the estimation of the available channel bandwidth. The video units are packetized into real-time transport protocol (RTP) packets, and they are delivered to the operating system's protocol stack (e.g. UDP/IP). The IP packets are fed into a FIFO

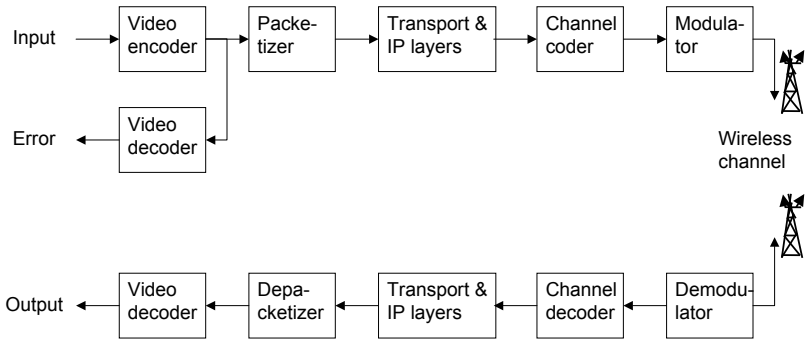


Figure 4: Components of a typical wireless video communication system.

output buffer before entering a packet lossy network, which can be the Internet, a wireless network, or a heterogeneous network. The network may have multiple channels (e.g., a wireless network) or paths (e.g., a network with path diversity), or support QoS (e.g., integrated services or differentiated services networks). Some packets may be dropped in the network due to congestion, or at the receiver because of excessive delay or unrecoverable bit error in a wireless network. To combat packet losses, parity check packets used for FEC may be generated in the application/transport layer. In addition, lost packets may be retransmitted if applicable. Packets that reach the receiver on time are buffered in the decoder buffer. The transport layer and application layer are responsible for depacketizing the received transport packets from the decoder buffer, channel decoding (if FEC is used), and forwarding the intact and recovered video packets to the video decoder. The video decoder then decompresses video packets and displays the resulting video frames in real-time (i.e., the video is displayed continuously without interruption at the decoder). The video decoder typically employs error detection and concealment techniques to mitigate the effects of packet loss.

2.3.1 Video Coding

The operation of the video encoder is to take raw video data and compress them so that temporal and spatial redundancy are reduced. Several successful standards have emerged which are basically separated into two main families of video compression standards: the H.26x family and the MPEG (Moving Picture Experts Group) family. Purpose of these standards is to address a wide range of issues such as bit rate, complexity, picture quality, and error resilience.

The H.26x family of standards, developed by the International Telecommunications Union-Telecommunications Sector (ITU-T), and aims at telecommunication applications and have developed from ISDN and T1/E1 service to embrace PSTN (Public Switched Telephone Network), mobile wireless networks, and LAN/Internet network delivery. The first standard of this family is H.261 ('90), which was designed for video communications at rates of 64kbps where the main requirement was low coding delay [19]. The H.263 standard ('95) was originally designed for very low bit rate applications, but eventually evolved into a significant improvement over H.261 at any bit rate [20]. As an extension of H.263, H.263+ and H.263++ ('97) [21] provide 12 new negotiable modes and additional features such as unrestricted motion vector mode, slice structure mode, scalability, etc. These modes and features further improve compression performance and error resilience. H.264, an on-going standard, aimed to achieve substantially higher video quality than the existing video standards at all bit rates [22]. Recently this standard was merged with the version 10 of MPEG-4 AVC (Advanced Video Coding).

The other important family of video coding standards is MPEG, developed by the MPEG group of the International Standards Organization (ISO). The first MPEG-1 standard was developed for CD-ROM applications with rates below 1.5 Mbps. MPEG-2 ('95) was designed for DVD, HDTV (High Definition Television) and digital satellites applications with rates between 2 and 20 Mbps. The next version was MPEG-4, which extends the basic hybrid-based video coding to object-based video, and it aims at low bit rate applications as well as interactive multimedia applications. The goal of MPEG-4 standard was to support new functionalities, such as improved coding efficiency, error robustness, and content-based

access, manipulation, and scalability.

The newest standard is H.264/AVC which aims to provide clean new standard that combines state-of-the-art compression technologies [45]. As we said, it is the result of the merger between the MPEG-4 group and the ITU H.26L committee in 2001, known as JVT (Joint Video Team), and is a logical extension to the previous standards adopted by the two groups. Thus, it is also called H.264, AVC or MPEG-4 part 10. The standardization of H.264/AVC is still ongoing. For an overview and comparison of the video standards, a detailed analysis can be found in [104]. MPEG-7 and MPEG-21 standards target the multimedia content description interface, which is different from traditional multimedia coding. It is important to note that all the standards are decoder standards, i.e., they standardize the syntax for the representation of the encoded bitstream and define the method for decoding process, but leave substantial flexibility in the design of the encoder. This limitation on the scope of standardization allows the maximal latitude of optimization for specific applications.

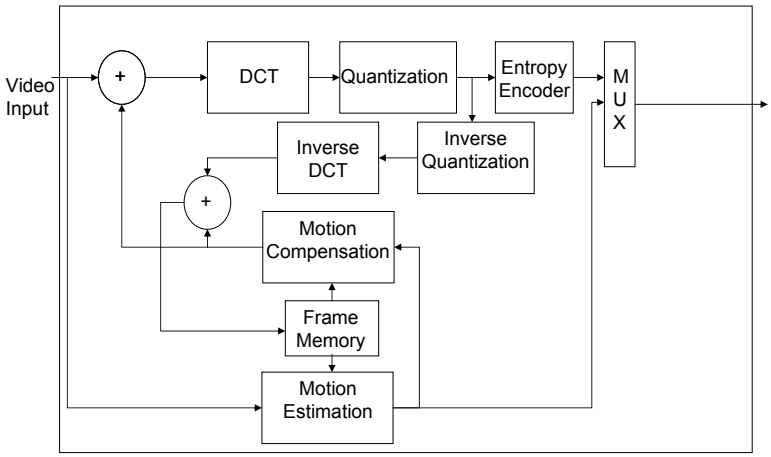


Figure 5: Block diagram of a typical hybrid-based video encoder.

Nevertheless, from the compression point of view, all the above mentioned video compression standards share the same block diagram, as shown in figure 5. This type of video

codec is called the block-based hybrid motion-compensated approach, where each video frame is presented in block-shaped units of associated luminance and chrominance samples (16×16 region) called macroblocks (MB). As shown in figure 5, the core of the encoder is motion compensated prediction. The first step in the motion compensation process is motion estimation, which aims to find the region from the previous frame that best matches each MB in the current frame. The offset between the MB and the prediction region is known as a motion vector. The motion vectors form a motion field, which is entropy encoded. The second step is motion compensation, where the reference frame is produced by applying the motion field to the previously reconstructed frame. The prediction error, known as the displaced frame difference, is obtained by subtracting the reference frame from the current frame. Following motion compensation, there are three major blocks to process, namely, transform, quantization, and entropy coding. The key reason in using transform is to decorrelate the data so that the associated energy in the transform domain is more compact and thus the resulting transform coefficients are easier to encode. The discrete cosine transform (DCT) is one of the most widely used transforms in image and video coding due to its high transform coding gain and low computational complexity. Quantization introduces loss of information, and is the primary source of actual compression. Quantized coefficients are entropy encoded, e.g. using Huffman or arithmetic coding. The input video frames, are divided into 8×8 pixel blocks, and DCT is then applied to each block, with resulting coefficients quantized. In these standards, a given MB can be intra-frame coded, inter-frame coded using motion compensated prediction, or simply replicated from the previously decoded frame. These prediction modes are denoted as intra, inter, and skip mode, respectively. Quantization and coding are performed differently for each MB according to its mode. Thus, the coding parameters for each MB are typically represented by its prediction mode and quantization parameter.

2.3.2 Video Streaming

In general, there are two ways to deliver pre-recorded video over a packet-oriented wireless network (and wired packet-switched network), and these are file download or streaming.

With file download, the entire video is downloaded to the users terminal before the playback commences. The video file is downloaded with a conventional reliable transport protocol, such as TCP. The advantage of file download is that it is relatively simple and ensures a high video quality. This is because losses on the wireless links are remedied by the reliable transport protocol and the playout does not commence until the video file is downloaded completely and without errors. The drawback of file download is the large response time, typically referred to as startup delay. The startup delay is the time from when the user requests the video until playback commences. Especially for large video files and small bandwidth wireless links, the start-up delay can be very large.

Pre-Recorded Video Streaming: With video streaming on the other hand, playback commences before the entire file is downloaded to the users terminal. In video streaming typically only a small part of the video ranging from a few video frames to several hundreds or thousands of frames (corresponding to video playback durations on the order of hundreds of milliseconds to several seconds or minutes) are downloaded before the streaming commences. The remaining part of the video is transmitted to the user while the video playback is in progress. One of the key tradeoffs in video streaming is between the start-up delay and the video quality. That is, the smaller the amount of the video that is downloaded before streaming commences, the more the continuous video playback relies on the timely delivery of the remaining video over the unreliable wireless links. The errors on the wireless links may compromise the quality of the delivered video in that only basic low quality (and low bit rate) video frames are delivered or some video frames are skipped entirely. Thus, video streaming gives the user shorter start-up delays at the expense of reduced video quality. The challenge of video streaming lies in keeping the quality degradation to a level that is hardly noticeable or tolerable while utilizing the wireless resources efficiently (i.e., supporting as many simultaneous streams as possible). The streaming service operates more or less in the same way as the video conferencing service. One of the differences are that the streaming service can be considered as an asymmetrical service because most of the information flows in one direction; from the server where the information is stored, to one or more clients. The requests from the client can be e.g. a request of retransmission due to an error in

transmission.

Live Video Streaming: The service of video conferencing requires that the media communication session is performed in real time. A video conferencing service is defined to be an audio visual conversational conference service providing two-way real-time transfer of voice and video between groups of users in two or more separate locations. Albeit the audio and video data are the most fundamental parts of the service, other types of data, such as still pictures, text or graphics may also be exchanged (ITU-T F.702, 1996). The 3G mobile system makes it possible to setup video conferencing services at low bitrates. Video conferencing is a service that contains continuous video and audio data should be delivered to an end user with a total latency of less than 200ms. This requirement requires a new set of mechanisms and algortihms that should be different from video streaming applications, and they should also be tailored to the specifications of the wireless radio access network [103].

CHAPTER III

TRANSPORT PROTOCOL MODELS FOR CBR AND VBR WORKLOADS

In this chapter we present analytical models that characterize TCP and TFRC throughput for different traffic workloads, namely CBR, VBR and bulk traffic in a hybrid wireless/wired network configuration. We compare the proposed model with existing TCP models, and we demonstrate through simulations the advantages of the proposed model. Subsequently, we analytically derive packet delivery bounds for TCP under the aforementioned workloads, and we present experimental results that highlight TCP's performance. Finally, with the help of the developed models and experimental results, we identify certain cases where TFRC is not able to support CBR and VBR workloads effectively.

3.1 Introduction

The transmission control protocol (TCP), has dominated the Internet traffic since its inception. Its widespread usage has spurred the development of several models that characterize its performance in terms of throughput, delay, and fairness [6, 5, 80, 22, 95, 115, 90]. The earlier TCP modeling efforts used continuous-time approaches [6], in order to obtain analytical formulas for the steady state throughput of a single flow. In subsequent work [80], researchers have proposed a widely-used model for TCP, that utilizes discrete-time Markov chains to characterize both the congestion window evolution and the achieved throughput. Models of throughput and latency for various flavors of TCP, i.e., Tahoe, Reno, and SACK, have been presented in a comprehensive work reported at [95]. A model that characterizes TCP window evolution as a set of stochastic differential equations can be found in [74], and other studies that model versions of TCP that do not enjoy wide deployment, i.e., TCP-Vegas, may be found at [91, 90]. Utilizing the analyses provided through these models, a new congestion-control protocol has emerged that is called TCP-friendly rate control

(TFRC) [46]. TFRC transmits non-TCP traffic at a rate similar to the rate that TCP would send data if the TCP flow was experiencing the same mean round-trip time and packet loss probability. This rate is predicted using the analytical models cited earlier [46]. It has been shown that the TFRC protocol achieves TCP-friendliness while it prevents unnecessary bandwidth fluctuations, by estimation of the packet loss rate and consequently the allowed output rate.

A common theme of the various performance models is that they characterize the protocol behavior for either bulk data flow or for short-lived flows. Arguably, the current and future Internet, is required to support various types of data, voice, and multimedia traffic workloads that may require a generalization of the assumptions that the previous models have been using. For example constant and variable bit-rate video (CBR & VBR) and VoIP, represent typical multimedia workloads which continue to gain importance in this context. Table 2 provides a summary of Internet workloads and their traffic characteristics. As seen in this table, Web-based interactions are usually modeled as Pareto ON/OFF sources that exhibit self-similar behavior [34, 11]. The behavior of VBR workloads is usually more complex and several modeling approaches have been presented in the literature [51].

There have been, to our knowledge, relatively few models capture the coupling between between transport protocol behavior under CBR and VBR workloads. Of specific interest in our work, are models that consider the case of heterogeneous traffic workloads, transported by a single end-to-end transport layer session. We propose and develop models for CBR and VBR workloads when they operate on top of TCP and TFRC, and we characterize the behavior of the TCP congestion window as a function of the workload. In this way more accurate estimates of the throughput may be made. Subsequently, we follow the same procedure in deriving the relationship between the actual rate, the allowed rate, and the used workload for TFRC.

3.2 Network Model and Assumptions

Figure 6 depicts the network model used in this study. This model consists of a traffic generator, that produces a workload according to a pre-defined model, and the TCP or

Table 2: Features of traffic characterization models.

Applications	**Quality requirements**	**Traffic type**
Voice over IP	Low delay/low jitter/no loss	CBR or VBR
Compressed video	Small delay/small loss	real-time VBR
Video streaming	Small delay/small loss	CBR
Real-time video streaming	Small delay/small jitter/small loss	VBR
WWW	High throughput/small loss	Self-similar
FTP	No loss/low delay	Self-similar

TFRC transport protocols, that absorb the load generated. We assume that one wired channel that is connected in tandem with a wireless channel, so that a hybrid network is created.

Workload and Network models: Concerning the traffic workload, we use a probabilistic model that considers the sources as a renewal reward process [89]. According to this model, at every round the workload generates an amount of packets that are absorbed by the protocol. The wired network is modeled as a two state Markov chain, also known as the Gilbert path model, that has been shown to predict the behavior the Internet packet loss quite well [118]. The network can either be in good or bad state, that translates into the successful delivery or loss of a specific packet. The wired packet loss probability is symbolized as p_w.

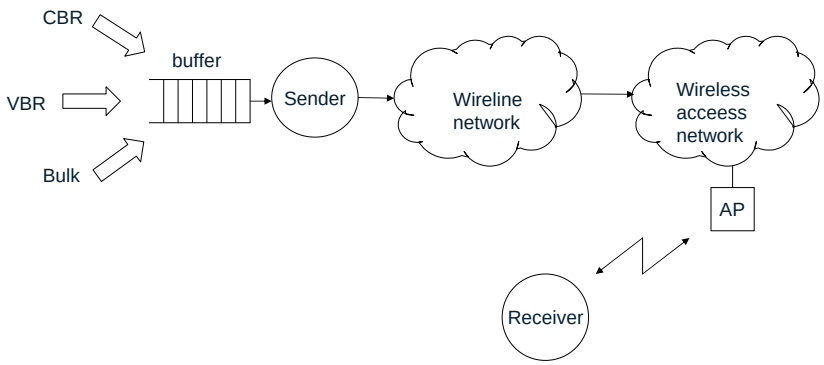


Figure 6: System model for heterogeneous traffic workloads.

Wireless Loss Model: We model the wireless link as a packet-erasure channel that is characterized by a bit error rate (BER). We assume a Rayleigh fading channel and so BER is calculated as [103]:

$$BER = \frac{1}{2}\left(1 - \sqrt{\frac{\alpha E_b}{N_0 + \alpha E_b}}\right) \tag{3}$$

Therefore the packet loss probability can be written as:

$$p_k = 1 - (1 - p_{e,k})^{B_k} \tag{4}$$

where $p_{e,k}$ is the BER after channel coding, and B_k is the packet size. In general, for wireless channels the probability of packet loss depends on the source coding and channel coding parameters, and of course the power level [103]. Now the probability of packet loss in the hybrid wireless/wired configuration, will be:

$$p_h = p_w + (1 - p_w)p_k \tag{5}$$

Protocol model: A TCP connection between two endpoints is defined by "rounds", similar to [62, 80], that have a duration of an RTT. During this round, TCP sends a burst of packets equal to the allowed window, and waits for acknowledgments. We name the number of RTT rounds that pass until there is a packet loss as an "NL round" (figure 7). Each rectangle in this figure, represents a packet that was sent during an RTT. Concerning the packet losses, we assume that they are correlated in each round implying that if a packet is lost, all the other packets in the same round are also lost [80]. Note that we make here the additional assumption that the end-to-end RTT remains stable for significantly large period. Otherwise the analysis would be complicated, and it would not lead to an analytically tractable model.

3.3 TCP Throughput Model for CBR Workload

An underlying assumption in existing models of transport protocols like TCP, is the existence of an infinite data backlog at the sender [80, 22, 95]. While this assumption may be valid for a majority of end-user current data-centric applications, it does not extend to constant bit rate (CBR) traffic flows. Multimedia traffic like CBR encoded video or VoIP flows,

represent common traffic patterns that may alter the behavior of TCP if used together, and motivate the analyses in this chapter. Consider now a CBR flow that is characterized by a rate of μ packets per second. This traffic workload can be considered as a renewal reward process, with a "reward" of μRTT packets that happens at every cycle that has a duration of an RTT. Figure 7 presents the packet-level behavior of this flow when TCP is used as the underlying transport protocol. In the absence of packet loss, TCP will continue increasing the size of congestion window for $1/b$ packets every RTT [99], where b is the number of packets acknowledged with a single ACK message. However, the CBR flow may not need to send as many packets as allowed by the congestion window, leaving *unused* packet slots in each RTT round. Furthermore, this also implies that the congestion window will stop increasing till the utilization of slots by packets reaches 100%.

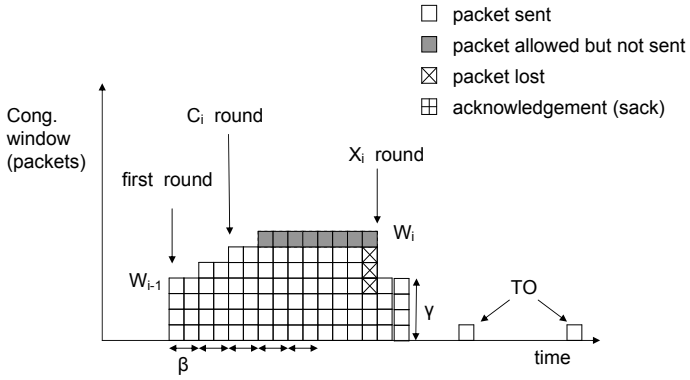


Figure 7: Packet-level TCP behavior of a CBR flow at the sender.

Model without timeouts: From figure 7, we can see that an NL round consists of $X_i b+1$ RTT rounds. The average number of packets sent, given that the l-th packet is lost, will depend on the packet loss probability p:

$$E[l] = \sum_{k=0}^{\infty} P\{l = k\} = \sum_{k=0}^{\infty} (1 - p_h)^k p = \frac{1}{p_h} \tag{6}$$

Given that the l-th packet is lost, $W-1$ more packets will be sent until the congestion window does not allow any more to be sent. Therefore, the total packets sent will be:

$$\begin{aligned} S^{CBR} &= l + W_i - 1 \Rightarrow \\ E[S^{CBR}] &= 1/p_h + E[W] - 1 \end{aligned} \tag{7}$$

Assume now that C_i is the RTT round for which the CBR flow rate is equal to the available TCP rate, and γ_i is the number of packets send in the final (X_i+1)-th RTT round. Consequently, from figure 7 we can write for the total number of packets send in an NL round:

$$S^{CBR} = \sum_{k=0}^{C_i/b-1} (\frac{W_{i-1}}{2} + k)b + \mu RTT(X_i - C_i) + \gamma \tag{8}$$

We now derive the distribution of the random variable C. This relies on the fact that C_i represents the RTT round for which the output rate μ of the CBR flow, is equal to the allowed TCP rate. Until that round, the window is increased for $1/b$ packets for each RTT round:

$$\begin{aligned} W_i &= W_{i-1}/2 + R_i/b - 1 \Rightarrow \\ E[W] &= 2(E[R]/b - 1) \end{aligned} \tag{9}$$

So for the C_i-th round for which the window stabilizes will be:

$$\begin{aligned} \mu RTT &= W_i = W_{i-1}/2 + C_i/b - 1 \Rightarrow \\ C_i &= b(\mu RTT + 1 - W_{i-1}/2) \end{aligned} \tag{10}$$

Finally when equations 8 and 10 are combined we get:

$$\begin{aligned} S_i^{CBR} &= \sum_{k=0}^{\mu RTT - W_{i-1}/2} (\frac{W_{i-1}}{2} + k)b + \gamma \\ &+ \mu RTT\Big(X_i - b(\mu RTT + 1 - W_{i-1}/2)\Big) \end{aligned} \tag{11}$$

In order to find the total number of rounds X_i included in an NL round as a function of W, we use equations 7, 9, and 11.

Timeout modeling: The next step towards the complete model, is the representation of Time-Outs (TOs). However, the used of a CBR workload does not alter the behavior of TCP either one TO happens or a series of them, when compared with any other workload. Therefore we can reuse results already available at the literature for accommodating this case. The average number of packets that will be sent when the sender suffers TOs will be [80]: $E[S^{TO}] = P_{TO}\frac{1}{1-p_h}$.

Therefore, the final expression for the throughput may be obtained as follows:

$$T^{CBR} = \frac{E[S^{CBR}] + P_{TO}\frac{1}{1-p_h}}{(E[X^{CBR}]+1)RTT + P_{TO}RTO_0\frac{1}{1-p_h}} \tag{12}$$

This formula, provides the TCP throughput as a function of the CBR rate μ, the packet loss rate of the hybrid network p_h, RTT, and acknowledgment ratio b. Note that if we assume that $\gamma_i = W_i/2$ then it has to be $\mu RTT > \beta_i$ otherwise the CBR flow rate requirement cannot be met after a decrease in the congestion window.

CBR plus Elastic Workload: Of interest is the case where an end-user application (e.g., a video conference or net-based meeting with voice and file transfers), generates different types of traffic workloads through the same end-to-end session [41]. The question that may come up is how the TCP throughput would be affected. Consider for example, the case that there is an additional bulk (or best effort) workload, that we would like to transport through the same TCP with CBR session. If there is such a bulk data flow, also fed to TCP, then the congestion window will increase after the CBR rate is satisfied in round C_i. Indeed, the bulk workload will start its transmission from round $C_i + 1$. Therefore, the total number of packets that will be sent by the bulk data flow in an NL round will be:

$$S^{BLK} = \sum_{j=C_i+1}^{X_i} (j - C_i)b \tag{13}$$

since for the first C rounds, the bulk data flow is idle because the bandwidth is assigned to the CBR flow. A prioritization method could be employed here by the application, effectively changing the ratio of the allocated bandwidth. So the throughput for TCP

would be:

$$T^{combo} = \frac{E[S^{BLK}] + E[S^{CBR}] + P_{TO}\frac{1}{1-p_h}}{(E[X^{CBR}]+1)RTT + P_{TO}RTO_0\frac{1}{1-p_h}} \tag{14}$$

3.4 TCP Throughput for VBR Workload

An alternative to CBR encoding for media traffic, is variable-bit-rate (VBR) encoding. With VBR, the quality of the media in terms of distortion does not suffer from large fluctuations contrary to the output rate which can exhibit significant variations. A number of research works related to the modeling of network behavior with VBR video streams, have been presented in the literature [17, 67, 70, 51, 61]. However, in this section we follow a novel approach, and design a joint model of TCP and a VBR workload of a typical video encoder.

Assume initially a VBR source that writes data (as shown in figure 6) to an intermediate buffer, from which then data can be consumed by TCP. We model this buffer in terms of rounds that have a duration r, similarly to the previously defined notation. Two operations can be executed to the buffer and these are to add and remove data. These operations take place at the start of each round. Therefore, at the round $j+1$, the amount of the buffer contents will be given by:

$$B_{j+1} = B_j + A_j - S_j \tag{15}$$

where S_j represents the data removed from the buffer by TCP, and A_j the packets added by the encoder. This equation indicates that the amount of data consumed by TCP at round $j+1$, depends on the current buffer occupancy B_j. This equation will eventually take the recursive form:

$$B_j = \sum_{k=0}^{j} A_k - \sum_{k=1}^{j} S_j \tag{16}$$

Now if $B_j \geq W_j$, then TCP can remove from the buffer a number of packets equal to the available window W_j, utilizing it thus 100%. On the other hand, if $B_j < W_j$, then TCP will consume all the data that exists in the buffer, leading thus to a buffer underflow. This means that the number of bytes sent in each round will be:

$$S_j = \begin{cases} W_j & \text{if } B_{j-1} > W_j \\ B_{j-1} & \text{if } B_{j-1} \leq W_j \end{cases} \tag{17}$$

Note that P{Buffer underflow}=P{$B_j < W_j$ }. So from equations 15 and 17 we can write:

$$S_{j+1} = \begin{cases} W_{j+1} & \text{if } B_j > W_{j+1} \\ B_{j-1} + A_{j-1} - S_{j-1} & \text{if } B_j \leq W_{j+1} \end{cases} \tag{18}$$

This means that the congestion window for the j-th RTT round of NL round i will be:

$$W_{i,j+1} = \begin{cases} W_{i,j} + 1/b & \text{if } B_j \geq W_j \\ W_{i,j} & \text{if } B_j < W_j \end{cases} \tag{19}$$

Figure 8 presents graphically how the behavior of the congestion window is affected, when a buffer underflow event happens. Now if we denote the number of RTT rounds for which $B_j \geq W_j$ as F, and the number of rounds that $B_j < W_j$ as G, then we have that $(G_i+F_i)b = X_i$. Consequently, the value W_i at the end of the NL round, will be $W_i = W_{i-1}/2 + bF_i$, making thus $E[W] = 2bF_i$. Since we do not know the value of the buffer size at the instant that the window remained unchanged, we assume that for the number of rounds G_i, the buffer will have an average value of B^{avg}. This assumption, makes the total number of packets sent in an NL round equal to:

$$S_i^{VBR} = S_i^F + S_i^G = \sum_{j=0}^{F} (W_{i-1}/2 + k)b + \sum_{j=0}^{G} B_i^{avg} \tag{20}$$

By using equation 7 which provides the estimate of the packet losses regardless of the workload, and equation 20, we obtain the duration X of an NL round with VBR workload.

The input of A_j bytes in a round r, depends of the VBR video traffic model used. We assume a simple stochastic model presented at [61], that characterizes the behavior of a video sequence that consists of I, P, and B frames [110]. In this model the distribution of the P and B frame sizes is modeled as i.i.d. log-normally distributed random variables with variance σ and mean μ for each of them. In addition, this model includes the size of an I frame as a function of the scene complexity, which is described by two random variables

J and L where J remains constant during a scene, while L is described by a normally distributed noise process. Given that the periods of I, P, and B frames are τ_I, τ_P, and τ_B respectively, and the number of P and B frames that correspond to an I frame are P_{num} and B_{num}, then the size in bytes that will be sent until time instant t is:

$$A_t = \sum_{l=0}^{t} \Big[(J_l + K_l) l\tau_I + \sum_{m=0}^{P_{num}} P_l^m (l\tau_I + \tau_P) + \sum_{m=0}^{B_{num}} B_l^m (l\tau_I + \tau_B) \Big] \quad (21)$$

Now because the period of I, P, and B frames does not match the RTT period, we have to find how many data are placed in the buffer during an RTT. This will be given by:

$$A_j \equiv A_t \quad \text{for which} \quad 1 < \lceil \frac{t\tau_x}{jRTT} \rceil \leq 2 \quad \text{with x} \in (I, P, B) \quad (22)$$

The average buffer occupancy is given by the average number of bytes added by the VBR workload minus the average number of packets sent by TCP. From equations 20 and 22, we

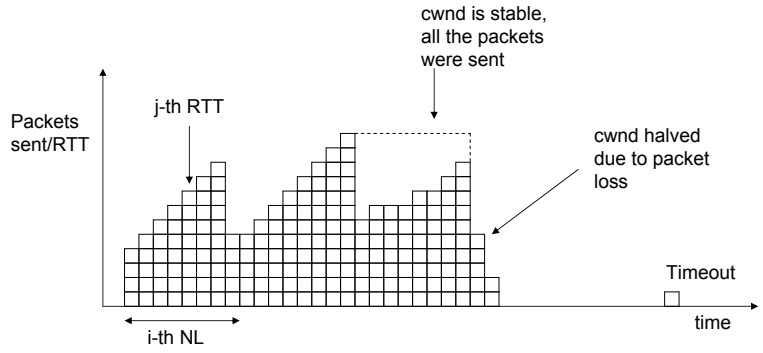


Figure 8: Congestion window evolution for TCP with a VBR flow.

can get the TCP throughput equation for the VBR workload:

$$
\begin{aligned}
E[S^{VBR}] &= bF\frac{W_{i-1}}{2} + b\frac{F(F+1)}{2} + GA_i \Rightarrow \\
T^{VBR} &= \frac{bFE[W]/2 + bF(F+1)/2 + GE[A]}{(Xb+1)RTT}
\end{aligned} \tag{23}
$$

3.5 TFRC Throughput and Latency Model

In this section we proceed with the characterization of the TCP-friendly rate control protocol (TFRC). We introduced TFRC is an equation-based rate control scheme, that is not a full-fledged transport protocol. In this study we consider TFRC to be implemented on top of UDP since UDP is protocol of choice for real-time media streaming applications. TFRC is using a closed form equation for TCP throughput in order to regulate the sender's output rate as a function of the packet loss rate p [46]:

$$
T = \frac{s}{RTT\sqrt{\frac{2p}{3}} + RTO_0(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{24}
$$

In this equation s is the packet size, RTT is the RTT estimate, and RTO_0 is the value of the retransmission timer. In addition TFRC follows a packet spacing algorithm at the sender:

$$
t_{inter} = \frac{s\sqrt{RTT_{cur}}}{T * M} \tag{25}
$$

where M is the average of the square roots of the RTTs calculated using an explicit window moving average (EWMA), and RTT_{cur} is the most recent RTT sample [46].

```
1: if p > 0 then
2:    x_calc = T(p, RTT, T_0)
3:    tfrc_x = max[min(x_calc, 2 * X_recv), s/tmbi]
4: else
5:    if tnow − tld ≥ RTT then
6:       tfrc_x = max(min(2 * tfrc_x, 2 * X_recv), s/RTT)
7:    end if
8:    tld = tnow
9: end if
```

Figure 9: The TFRC rate estimation algorithm.

Equation 1 that we presented in chapter 2 of this dissertation, does not represent the actual TFRC sending rate but only an upper bound for it. The actual output rate of TFRC is calculated using the algorithm in figure 9. In this algorithm, s represents the packet size, tld is time when the rate was last doubled, and $tmbi$ is the maximum back-off time (64 seconds by default). If p is zero, no packet loss has yet been "seen" by the sender and in this phase it emulates the slow start algorithm of TCP by doubling the transmission rate every RTT. The condition, $tnow - tld \geq RTT$, assures that the rate is not doubled more than once during an RTT, similar to TCP [99]. $Xrecv$ is the average receive rate at the receiver. This value is calculated in appendix A (equation 121). For calculating the average sender rate, we consider the case where TFRC is in congestion avoidance and thus $p > 0$. If we combine equations 1 and 121, we obtain the total number of packets sent by TFRC in a single NL round:

$$E[S] = \max\left(2\min(T, \frac{2E[R]}{RTT}), \frac{s}{tmbi}\right) \tag{26}$$

In the next subsection we will see how this equations can be utilized in order to model the transport of more complex workloads.

If we assume a constant value for RTT, then the only parameter that could affect the TFRC rate estimate is the packet loss rate p. If TFRC receives reports every δRTT seconds, then in between these epochs, the output rate will be stable. The inter-packet spacing will also be fixed during this periods. So the average number of packets lost, when m packets are sent, will also be given by equation 6. The VBR traffic and buffer model developed earlier for TCP, can also be used for TFRC since the buffer state is independent of the protocol behavior. So in the j-th RTT round TFRC will send:

$$S_j^{TFRC} = \begin{cases} T_j^{TFRC} RTT & \text{if } B_{j-1} > T_j^{TFRC} \\ B_{j-1} & \text{if } B_{j-1} \leq T_j^{TFRC} \end{cases} \tag{27}$$

As mentioned before, till the next time p is reported to the TFRC-based sender, the total number of packets sent is given by:

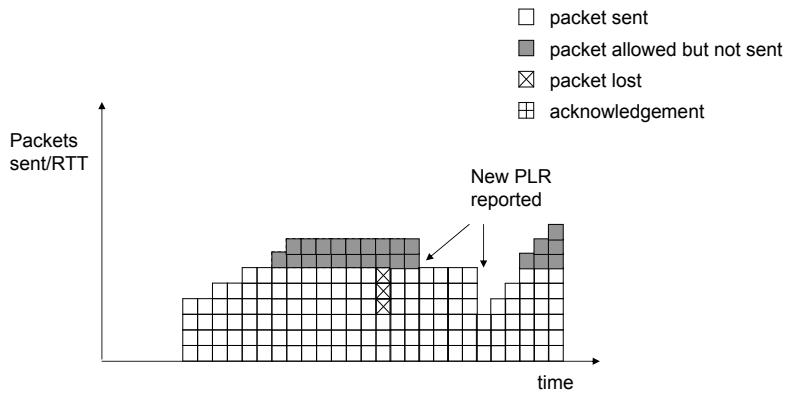


Figure 10: Packet-level TFRC behavior of a CBR flow at the sender.

$$S^{VBR} = S_F + S_G = \sum_{j=0}^{F} (T_j^{TFRC} RTT) + \sum_{j=0}^{G} B_{avg} \tag{28}$$

Modeling TFRC with a CBR flow is similar to the case of the TCP, as the number of rounds F can be calculated as similar with before. Consequently we have for TFRC:

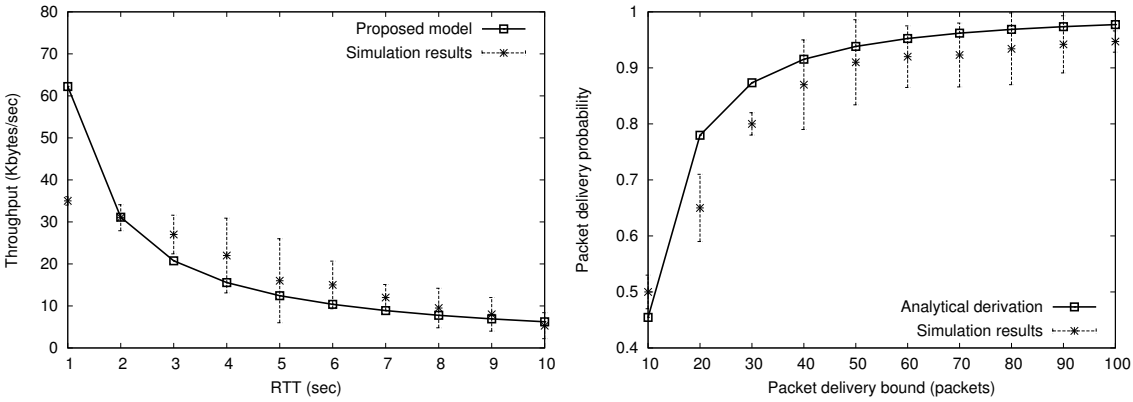
$$\begin{aligned} T^{VBR} &= \frac{E[S^{VBR}]}{XRTT} \Rightarrow \\ T^{VBR} &= \frac{F \times T^{TFRC} RTT + GE[A]}{b(G+F)RTT} \end{aligned} \tag{29}$$

Note that in the case of TFRC $X = \delta$, since each NL round is defined as the duration between the time instants that the sender received feedback reports.

3.6 Numerical Results and Simulations

We used a server/client configuration with a bottleneck link between two routers and the Reno version of TCP for our simulations. The results were obtained from 100 runs of the ns-2 simulator [77], that simulated a duration of 500 seconds in the scenario. The bottleneck link was varied from 1.5 to 5 Mbps, while the latency was set to 15ms, and buffer sizes were

set so that no packet loss takes place due to overflow. Packet losses were generated as we discussed earlier.

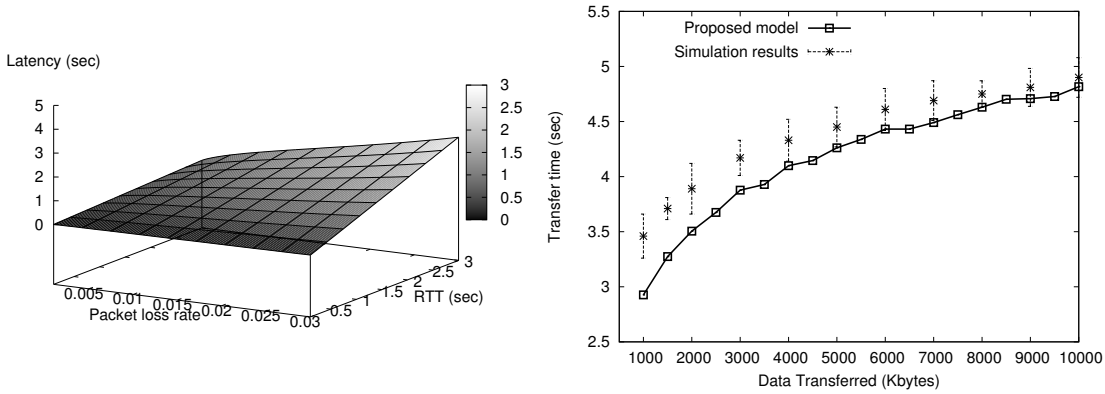


(a) Goodput at the receiver versus the RTT

(b) Packet delivery probability versus the bound at the receiver

Figure 11: Simulations for TFRC receiver model validation. Parameters: $p = 0.02$, $RTO_0 = 1$ sec, $s = 1500$ bytes.

3.6.1 TFRC with Elastic Traffic (Wireline)



(a) Model predicted latency for the transfer of 1MB with TFRC

(b) Latency vs. data transferred with $RTT = 1sec$ and $plr = 0.01$

Figure 12: Analytical results and simulations for TFRC latency model validation. Parameters: $RTT_0 = 1$ sec, $s = 1500$ bytes.

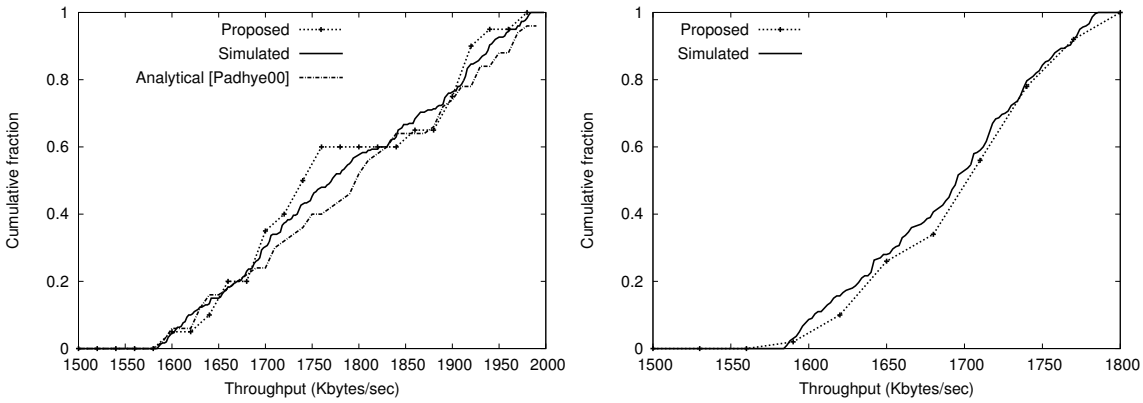
Results for the TFRC model are depicted in figure 11. The simulation results in this figure represent the average percentage of packets delivered for 100 simulation runs. A first important observation in figure 11(a), is that TFRC can meet the packet delivery bounds

easier than TCP due to its slower reaction to congestion events. Figure 11(b) also validates this fact since we see that TFRC can have higher probability of successful packet delivery, for a lower packet delivery bound. This means that tighter packet delivery bounds can be met with higher probability from TFRC than from TCP. Further results for TFRC are presented in figure 12. More specifically, figure 12(a) presents the average latency as a function of RTT and the packet loss probability. It is interesting to note that TFRC latency, is characterized by a linear increase as RTT and packet loss probability increase. However, the increase of the end-to-end latency as a function of the number of transferred bytes (figure 12(b)), does not follow a linear pattern. These results were obtained for fixed RTT and packet loss probability values that are given in figure 12(b). In this figure we can see that initially the results of the proposed model diverge from the simulation, but as the size of the transferred data is increasing, the model approaches closer the real measurements. This behavior is observed because our model does not include the slow-start behavior of TCP, which has more dominant effect for smaller size transfers.

3.6.2 TCP and TFRC with CBR and VBR Traffic (Wireline)

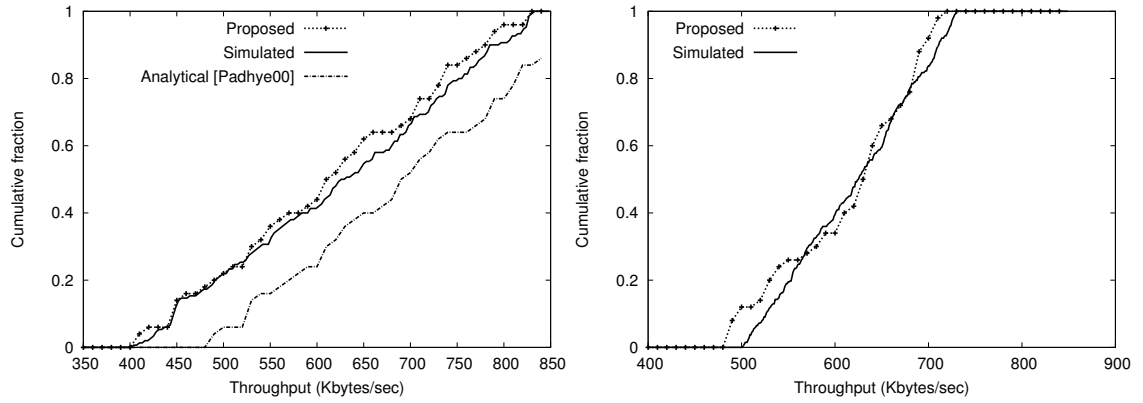
For testing the VBR TCP model, we set the parameters of two popular video sequences namely CONTAINER and COASTGUARD [94]. Figure 13(a) illustrates the cumulative fraction of the throughput when the VBR flow was used as input to TCP. In the bottleneck link we generated losses with average packet loss probability of $p = 0.001$. The VBR parameters that were explained before, were set as follows: $\mu_J = 5.9$, $\sigma_J = 0.48$, $\mu_B = 3.9$, $\sigma_B = 0.27$, $\mu_P = 4.8$, $\sigma_P = 0.64$. The proposed model can predict the achieved throughput fairly well since it can accommodate the rate fluctuations in the input. Results for the TCP model presented at [80], are also shown in the same figure. Their approach provides a slightly more optimistic throughput estimate since the occupancy of the buffer was set to F=90%, resulting into a small number of rounds where the congestion window is not increased. Results for TFRC are shown in figure 13(b). We can see that TFRC can achieve more stable throughput since it only fluctuates between 1600 and 1800 Kbytes/sec. Also the model can predict very close the actual throughput for the same buffer occupancy of

90%.



(a) TCP (b) TFRC

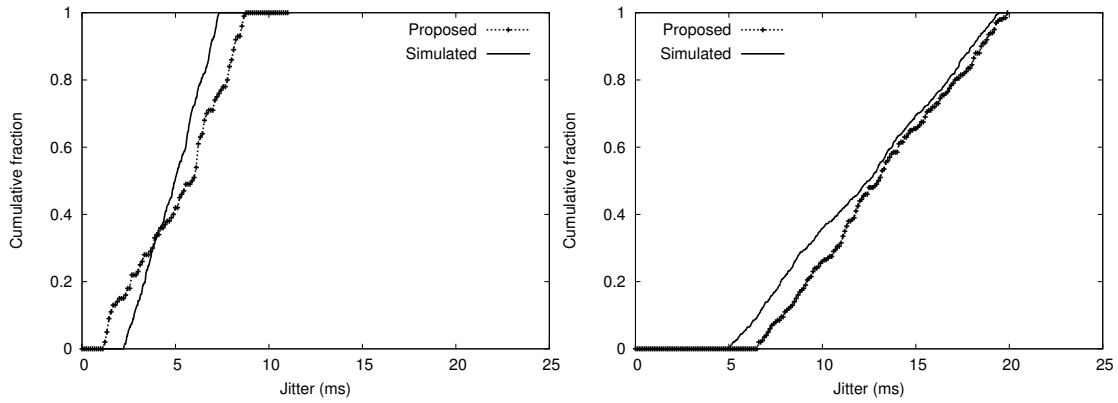
Figure 13: Cumulative fraction of the end-to-end throughput for VBR workload with buffer occupancy $F = 90\%$. Parameters: $RTO_0 = 200$ ms, $W_0 = 1$ segment, $W_{max} = 6$ MB, $RTT_0 = 1$ sec, $s = 1500$ bytes.



(a) TCP (b) TFRC

Figure 14: Cumulative fraction of the end-to-end throughput for VBR workload with buffer occupancy $F = 50\%$. Parameters: $RTO_0 = 200$ ms, $W_0 = 1$ segment, $W_{max} = 6$ MB, $RTT_0 = 1$ sec, $s = 1500$ bytes.

The behavior of both TCP and TFRC with lighter VBR workloads is quite interesting. Figure 14 presents results for a VBR workload with an average buffer occupancy of 50%. For TCP, this lighter load, leads to an over-estimation of the actual throughput when the existing TCP models are used 14(a). However, the ability to communicate the precise input load to the TCP model leads to a very good estimate for the proposed model, also depicted



(a) TFRC jitter with bulk traffic workload

(b) TFRC jitter with VBR workload and buffer occupancy F=50%

Figure 15: Cumulative fraction of the end-to-end jitter for TFRC. Parameters: $RTT_0 = 1$ sec, $s = 1500$ bytes.

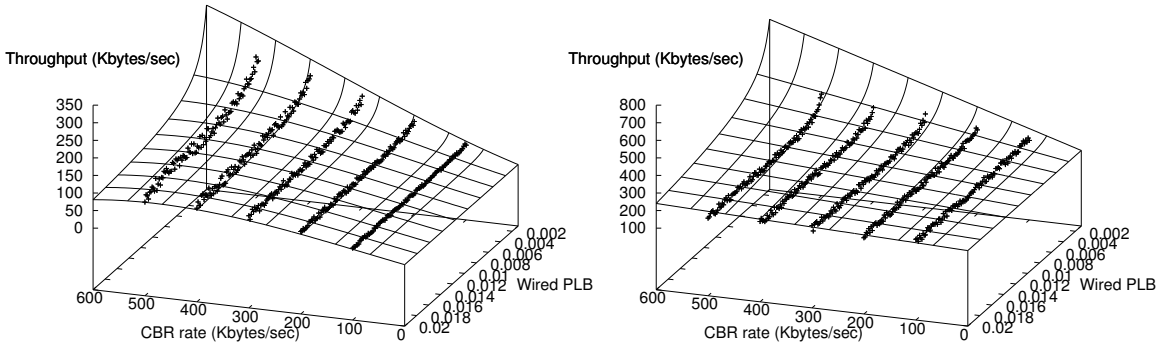
in figure 14(a). The same results ar observed for TFRC in figure 14(b).

A very interesting experiment is presented next. We wanted to evaluate TFRC's ability to provide low jitter transport service. The expected behavior is to observe low end-to-end jitter due to the smooth rate control algorithm that TFRC employs. Figure 15(a) validates the expected TFRC behavior, when a bulk traffic workload is considered. However, the interesting result is in figure 15(b), where jitter is shown for a VBR workload with an input buffer occupancy of $F = 50\%$. The actual jitter is different, and actually higher for the same network conditions as before. The proposed model can predict this unexpected TFRC behavior for a different workload. What it actually happens, is that the protocol commits a smaller amount of data in the network when compared with a bulk workload. This situation leaves part of nominal bandwidth unused, which is subsequently captured by the other TCP cross traffic. This essentially means that even if TFRC uses smaller part of the bandwidth it will have problem maintaining a stable output rate, and therefore end-to-end jitter.

3.6.3 TCP and TFRC with CBR and VBR Traffic (Wireless)

Evaluating the protocol's performance under a hybrid wireless/wired network setup revealed very interesting results concerning the TCP and TFRC's ability to provide multimedia support in these scenarios. We selected a fixed value for the wireless packet loss probability

for this set of experiments. The reason behind this decision, is because wireless networks usually heavily employ local retransmissions that lead to a fairly low packet loss rate. However, this mechanism usually lead leads to RTT fluctuations, that may affect the model predictions.



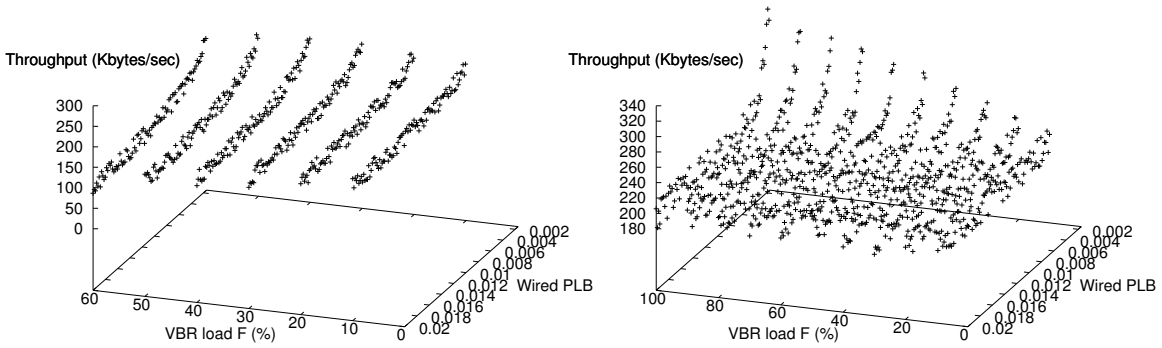
(a) Results for TCP

(b) Results for TFRC

Figure 16: Throughput as a function of the CBR rate μ (x-axis) and wireless/wired packet loss ratio (y-axis). $p_k = 0.001$ and p_w varies from 0.001 to 0.02.

The results for TCP in figure 16(a) indicate a rapid decrease in throughput as a function of the wired packet loss probability. Also important is to note that the CBR load does not heavily affect the achieved throughput. This behavior is something to be expected when the packet loss probability is relatively high, and so it it dominates the protocol behavior. However, for the case of TFRC, we see in figure 16(b) that the throughput is reduced sharper as the workload rate is reduced, regardless of the packet loss probability. This sensitivity of TFRC has its root in the slow-responsive nature of the protocol. When TFRC does not commit data to the network, and it still suffers losses, it reduces the rate even lower that the nominal bandwidth, resulting into severe throughput degradation.

Results for VBR workload are shown in figure 17. For clarity purposes we only show the data points obtained from simulations. What we hope to see here again, is the effect of asymmetric packet losses on the throughput of typical VBR multimedia workload. The results for TCP in figure 17(a) indicate that a VBR workload that provides data to TCP



(a) Results for TCP

(b) Results for TFRC

Figure 17: Throughput as a function of the VBR load F.

between 20-50% of the allowed rate, it can achieve fairly stable throughput that moderately depends on the packet loss rate. However, as the VBR load is increased, this behavior changes since the workload behaves closer to an elastic workload with bulk data traffic. This event contributes to the increase of the effect that cross traffic has on the TCP throughput. Results for TFRC indicate a similar behavior with the case of the CBR workload. This means that the unpredictable arrivals of the incoming workload packets at the protocol sender, do not affect TFRC behavior as much as just the reduced workload. So for either CBR or VBR workloads with nearly 50% load of the available bandwidth, TFRC receives even lower part of the bandwidth and it cannot maintain a constant rate.

3.7 Conclusions

In this chapter we presented analytical models that characterize TCP and TFRC throughput for different traffic workloads, namely CBR, VBR and bulk traffic in a wireless/wired network setup. The first important conclusion that we draw from the analysis in this chapter, is that the assumption of flows with an infinite data backlog, may significantly affect the TCP throughput estimate in case of CBR and VBR workloads. We demonstrated that with our model, these predictions can be more accurate, leading to a better understanding of the protocol and workload interactions. Therefore, when performance is evaluated, someone should try to correlate carefully the transport protocol in use, with the actual workload.

We identified TFRC's inability to provide high throughput service when the traffic workload is characterized by large rate variations (e.g. VBR). This means that a number of additional factors have to be considered before deploying the protocol for a media application. For wireless scenarios, the proposed model does not differentiate significantly. Even, the asymmetry in the packet loss probability across the wireline wireless networks does not significantly the throughput since both TCP and TFRC experience the aggregate packet loss.

CHAPTER IV

RATE-DISTORTION OPTIMIZED UNICAST VIDEO STREAMING WITH TCP

In this chapter we present an analytical study that characterizes the performance of video streaming with the transmission control protocol (TCP). First, we develop an analytical model of the expected video distortion at the decoder with respect to the TCP parameters, channel state, and error concealment method at the receiver. Based on this model, we propose an algorithm for rate distortion optimized mode selection (RDOMS) for video streaming with TCP. Experimental results for real-time video streaming depict improvement in PSNR in the range of 2 db over currently proposed TCP-based streaming mechanisms. Our next contribution is the development of a joint model of the TCP protocol, and the playback buffer at the receiver. Based on this model, we derive the optimal playback rate at the decoder. Subsequently, based on the previously developed models, we propose an algorithm, for rate distortion optimized packet scheduling with TCP. Our results show an additional improvement of nearly one db, when packet scheduling is applied together with the RDOMS algorithm.

4.1 Introduction

Video streaming represents a very popular application for IP networks and especially the Internet. However, the TCP protocol that dominates Internet traffic [38], is considered unsuitable for video streaming applications. The main reasons are the rapid throughput fluctuations and the reliability mechanism which incurs additional delays [110]. Therefore, it is generally believed that the transport protocol of choice for video streaming should be UDP, on top of which several application specific mechanisms can be built [110]. However, the absence of congestion control from UDP can cause performance deterioration for TCP-based applications if wide-scale deployment takes place in the Internet [104, 52]. This is the

reason behind IETF's effort to design a new rate control protocol that realizes congestion control compatible with TCP while it allows smoother throughput fluctuations [46].

Despite these efforts however, the majority of commercial IP-based video streaming systems usually employ TCP for transport layer services [87, 111, 85]. The widespread use of TCP, has stimulated research for the development of mechanisms that facilitate video streaming with TCP. In [60], the authors evaluate multimedia streaming using TCP, by noting that buffering at the client can handle the retransmission delays and the congestion control induced throughput variations of TCP. Another approach reported in [72], attempts to provide a nearly CBR channel to the streaming flow that is using TCP, through prioritization over other flows at the receiver. The limitation with this approach is that it assumes that congestion induced bandwidth fluctuations are due to the "last mile" connection which is rarely the case [38]. Other mechanisms like time-lined TCP [76], propose the realization of streaming over TCP by allowing the operating system to control the transmission of data that have strict deadlines. A similar approach to the previous one, can be found in [68], where the proposed protocol TCP-RTM requires significant modifications both to the TCP sender and receiver. A receiver-driven technique for video streaming with TCP, introduced the idea of receiver-based delay control [49], in which receivers delay TCP acknowledgments based on feedback from routers. Nevertheless, the need for infrastructure modifications renders this mechanism impractical for end-to-end video streaming applications. In another insteresting recent approach, the authors introduced a new client-driven RTO estimation algorithm which has as objective of identifying lost packets as quickly as possible [12]. Subsequently, the authors combine their algorithm with a simple retransmission-based error-control method in order to analyze the impact of the RTO estimation for video streaming.

In summary, one common deficiency of the above mechanisms is that they consider modifications and enhancements to TCP or the infrastructure, and ignore the nature of the content which is a video stream. While the previously mentioned optimizations may be the only option for pre-recorded video streaming, in the case of real-time video streaming, where the video encoder can accept feedback during the encoding process, a number of techniques can be applied in order to improve the quality of the delivered video. To exploit

this potential, the video coding community has developed several mechanisms for network adaptive video streaming. More specifically there are several methods that attempt to achieve network friendliness through adaptation at the video encoder [113]. For example an interesting approach is rate-distortion optimization (RDO) [69]. The advantage of this approach is that it can accommodate rate fluctuations with appropriate modification of the quantization parameter at the encoder. However, large variations in the received video quality and significant processing overhead are two of its disadvantages. Other approaches use RD metrics in order to select the optimal encoding mode at the sender [112]. Further improvements led to RD algorithms that attempt to meet multiple packet deadlines by using back-channel messages [53]. A complete and general framework for the RD optimized packet scheduling has been presented at [33, 32]. The authors developed a general framework for estimating the end-to-end distortion of video for various streaming scenarios. They propose a heuristic algorithm for finding a suboptimal scheduling policy that optimizes the receiver distortion. Several more sophisticated algorithms have been developed since [117, 65, 106, 7, 25]. For example in [117], a new scheme for RD optimized streaming for MPEG-4 layered video that considers the use of unequal error protection through FEC. Other works for RD optimized streaming include [73, 116], in which the authors proposed new heuristic algorithms for calculating the expected distortion in real-time.

4.2 System Overview

In figure 18 we show a simplified block diagram of the proposed video streaming system. Similar to the most widely used media streaming applications [111, 85], the proposed end-to-end media streaming system based on TCP, is using a small startup delay Δ before the video playback starts, while the server sends video packets which are stored at the client playback buffer. While video palyback starts, the server continues to send new data to the end of the client playback buffer, while the decoder keeps consuming the data available at the start of the buffer.

During the video streaming session, the playback delay fluctuates as a function of varying network conditions, such as the available bandwidth, and the end-to-end transmission delay.

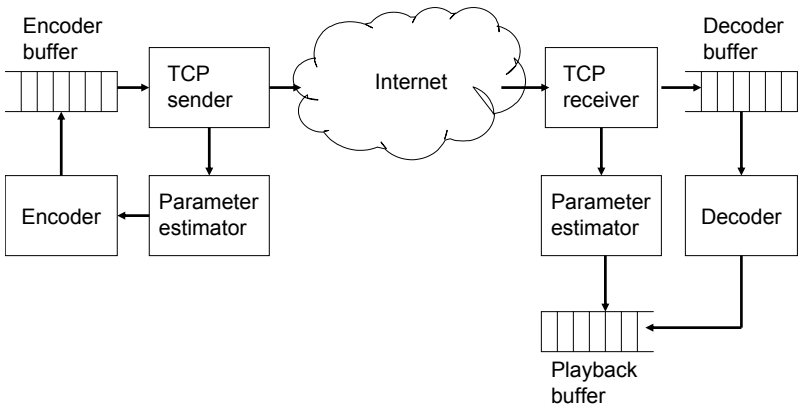


Figure 18: Proposed real-time media streaming architecture based on TCP.

The use of an adaptive playback buffer is used at the receiver in order to accommodate network jitter. The problem that is formed, is finding the minimum possible buffer size at the receiver so that network jitter is smoothed. On the other hand, the video encoder is using the distortion characteristics of the streamed video in order to maximize the video quality at the receiver. The video streaming application evaluates the end-to-end distortion of the video, as a function of the distortion of the individual macroblocks, the expected end-to-end latency, the error concealment at the decoder, and the state of the transmission path. This results in better scheduling policies that maximize the expected quality at the receiver.

Our study jointly optimizes the previous parameters, and it also adds another degree of flexibility at the encoder by considering the effect of the specific algorithms a transport protocol is using. Another important feature of our approach, is that it requires no modifications to the protocol stack, and especially TCP, making it thus a practical approach to for video streaming. Our belief is that the area of video coding and transport protocols should be brought together by more careful joint study of the video coding algorithms

and the transport protocols. This work represents a step toward this direction with the consideration of the TCP protocol.

4.3 Path Model and Packet Loss Estimation

The network model adopted, captures the behavior of an end-to-end path, as a three-state Markov chain. Generally, the two-state Markov chain has been shown to predict fairly well the behavior of the Internet with respect to packet loss behavior [8]. However, a higher order Markov chain is selected, since we want to define three channel states that determine the fate of a packet, namely received (R), lost (L), and delayed due to TO or TD (D). This distinction is made because we want to separate the channel induced packet loss, and the delayed packets which are delayed due to TCP retransmissions. According to the expected delay of a specific TCP packet, the contribution of the video packet that this packet is carrying to the expected distortion changes. This intuitive idea is the one that guided the development of this comprehensive protocol/distortion model. Therefore, for this path model, the transition matrix between the three states is given by:

$$A = \begin{pmatrix} \pi_{RR} & \pi_{RL} & \pi_{RD} \\ \pi_{LR} & \pi_{LL} & \pi_{LD} \\ \pi_{DR} & \pi_{DL} & \pi_{DD} \end{pmatrix} \tag{30}$$

The notation π_{xy}, symbolizes the transition from state x to y. Now if we want to capture the probability to transition from one state to another after n successfully delivered packets, we can re-write the previous matrix as:

$$A^n = \begin{pmatrix} \pi_{RR}^n & \pi_{RL}^n & \pi_{RD}^n \\ \pi_{LR}^n & \pi_{LL}^n & \pi_{LD}^n \\ \pi_{DR}^n & \pi_{DL}^n & \pi_{DD}^n \end{pmatrix} \tag{31}$$

The probability for a packet to be delayed after n packets were sent, and were either received lost or delayed, is equal to:

$$P_D = \frac{\pi_{RD} + \pi_{LD} + \pi_{DD}}{\pi_{RD} + \pi_{DR} + \pi_{LD} + \pi_{DL} + \pi_{DD}} \tag{32}$$

The transition probabilities are calculated using maximum likelihood estimators [18]: $\hat{\pi}_{RD} = \frac{n_{RD}}{n_R}$, where n_{RD} is the number of times in the ACK messages that D follows R and n_{DR} is the number of times R follows D. On the other hand n_R, is the number of received packets (R), regardless of the previous state. Concerning the calculation of n_{RL}, n_{LR}, n_{LD}, n_{DL}, we follow the same procedure. Note that for the receiver, it is relatively easy to identify and classify a packet into one of the three aforementioned states, since it knows if the packet was delayed or not.

Concerning the incurred latency for the delivery of a TCP packet, it can be estimated after we delve into the inner mechanisms of TCP. By doing so, we can identify the following three cases: The packet was received correctly the first time it was sent, the packet was received from a retransmission that was triggered by three duplicate (TD) acknowledgments, or the packet was received from a retransmission after an RTO expiration at the sender (TO). We will calculate the probability that a packet that was sent at time t_s, has delayed for time L in the network, and this resulted into the missed deadline t_d. In the previous three cases, the above statement can be expressed as $P\{L+t_s > t_d\} = 1-P\{L+t_s \leq t_d\} = 1 - F_L(t_d)$, where $F_L(t)$ is the cumulative distribution function of the end-to-end latency. Therefore, we have to calculate the latency distribution for the above three cases.

When the only source of latency is the network, no TCP mechanism has to be modeled, since a packet is suffering only the network induced delay. Therefore, we model the distribution of the the one way network induced latency f_{L_N}, as a shifted Gamma distribution with probability density function [25, 64]:

$$f_{L_N}(t) = \begin{cases} \frac{\lambda e^{-\lambda t}(\lambda t)^{\nu-1}}{(\nu-1)!} & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \tag{33}$$

with mean ν/λ, and variance ν/λ^2.

In the second case where the TCP sender receives three duplicate acknowledgments, due to a lost packet, it fast retransmits the packet that the duplicate ACKs indicate as missing. The latency incurred due to a fast retransmission will be equal to the probability that a TD event takes place $(1 - P_{TO})$, where P_{TO} is the RTO expiration probability, times the RTT of the connection (duplicate ACK notification plus retransmission). This latency will

be given by: $L_{TD} = (1 - P_{TO})RTT$. The latency for a retransmitted packet that comes from an RTO expiration will be equal to $L_{TO} = P_{TO}Z^{TO}$ where Z^{TO} is the duration of the timeouts. So the value $P\{L_N + L_{TD} + t_s > t_d\} \bigcup P\{L_N + L_{TO} + t_s > t_d\}$, will give the probability that a packet has missed the playback deadline due to a fast retransmission or a timeout at the TCP sender. Therefore, the total probability for the packet to be late is:

$$\begin{aligned} P_D^{TCP} &= P\{L_N + t_s > t_d\} \\ &\bigcup P\{L_N + L_{TO} + t_s > t_d | \text{lost and caused TO}\} \\ &\bigcup P\{L_N + L_{TD} + t_s > t_d | \text{lost and caused TD}\} \end{aligned} \tag{34}$$

This equation, captures the fundamental mechanisms that can be a source of delay for a TCP based session. We will revisit this equation in section 4.5, were we will see how we can use a version of it in order to drive on-the-fly decisions concerning the video streaming process. In the next section, we move one step further, and see how the knowledge of the expected latency for classes of packets, can be used in a distortion model.

4.4 Analytical Model of the Expected Video Decoder Distortion

In this section we derive an analytical model concerning the expected distortion at the decoder as a function of the channel state, TCP introduced latency, and the error concealment method used at the decoder.

Let M_i^n be the coded MB at location i of frame n, and let also $M_i^n \in X_k$ symbolize the fact that a coded macroblock is contained in network packet X_k. Let also f denote the pixel value at the encoder, and $\tilde{f}$ the reconstructed value at the receiver, and $\hat{f}$ the encoder estimation of the reconstructed pixel value at the decoder. Figure 19 depicts clearly this arrangement. If we denote as η_i^n as the last packet used to packetize the MB M_i^n and as K the number of packets that packetize the first I frame of the series (i.e. Ψ), then the probabilities for MB i that belongs to frame n to be lost $\tilde{\pi}_L^{(i,n)}$, received $\tilde{\pi}_R^{(i,n)}$, and delayed $\tilde{\pi}_D^{(i,n)}$ are given as follows, if we assume that the first intra frame is always received:

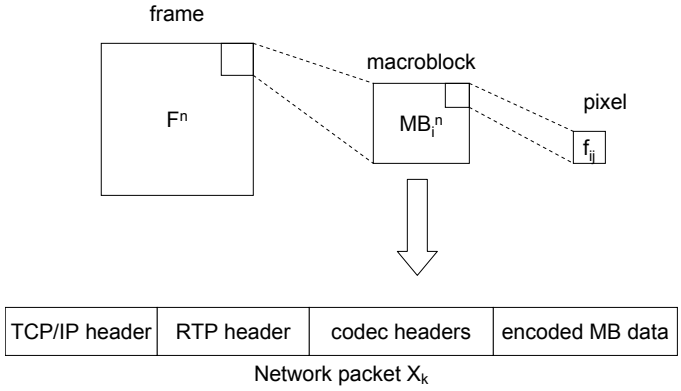


Figure 19: Packetization of encoded macroblocks

$$\tilde{\pi}_R = \begin{cases} 1 & \text{if } n = 0 \\ \pi_{RR}^{\eta_i^n - K - 1} & \text{if } n > 0 \end{cases} \tag{35}$$

$$\tilde{\pi}_L = \begin{cases} 0 & \text{if } n = 0 \\ \pi_{RL}^{\eta_i^n - K - 1} & \text{if } n > 0 \end{cases} \tag{36}$$

$$\tilde{\pi}_D = \begin{cases} 0 & \text{if } n = 0 \\ \pi_{RD}^{\eta_i^n - K - 1} & \text{if } n > 0 \end{cases} \tag{37}$$

The above equations mean that the probability $\tilde{\pi}_R^{(i,n)}$ to receive MB i from frame n correctly, equals to the probability of receiving successively $\eta_i^n - K - 1$ packets correctly. For the calculation of $\tilde{\pi}_L^{(i,n)}$, we have to consider the case of loosing the last of the $\eta_i^n - K - 1$ packets, given that the MB $\eta_i^n - K - 2$ has been successfully delivered. In addition for $\tilde{\pi}_{RL}$ and $\tilde{\pi}_{LL}$, we have:

$$\tilde{\pi}_{RL} = \begin{cases} 0 & \text{if } n = 0 \\ \pi_{RR}^{(\eta_{i-1}^n - K + 1)} \pi_{RL}^{(\eta_i^n - \eta_{i-1}^n)} & \text{if } n > 0 \end{cases} \tag{38}$$

$$\tilde{\pi}_{RD} = \begin{cases} 0 & \text{if } n = 0 \\ \pi_{RR}^{(\eta_{i-1}^n - K+1)} \pi_{RD}^{(\eta_i^n - \eta_{i-1}^n)} & \text{if } n > 0 \end{cases} \tag{39}$$

$$\tilde{\pi}_{LL} = \begin{cases} 0 & \text{if } n = 0 \\ \pi_{RL}^{(\eta_{i-1}^n - K+1)} \pi_{LL}^{(\eta_i^n - \eta_{i-1}^n)} & \text{if } n > 0 \end{cases} \tag{40}$$

$$\tilde{\pi}_{LD} = \begin{cases} 0 & \text{if } n = 0 \\ \pi_{RL}^{(\eta_{i-1}^n - K+1)} \pi_{LD}^{(\eta_i^n - \eta_{i-1}^n)} & \text{if } n > 0 \end{cases} \tag{41}$$

The above equations can be interpreted as follows: The probability to loose MB η_i^n, precisely after a successful MB delivery (π_{RL}), is given as the probability of receiving the previous MB $(i-1)$ correctly with probability $\pi_{RR}^{(\eta_{i-1}^n - K+1)}$, times the probability to have another chain of RRRRL... events which depends on the id of the packets used for MB i and $i-1$, and their relative distance $(\eta_i^n - \eta_{i-1}^n)$ in the network packet. So this means that the probability for a MB to be delayed, that was calculated in equation 32, if we account for the packetization that we analyzed in this section, will be:

$$P_D = \frac{\tilde{\pi}_{RD} + \tilde{\pi}_{LD} + \tilde{\pi}_{DD}}{\tilde{\pi}_{RD} + \tilde{\pi}_{DR} + \tilde{\pi}_{LD} + \tilde{\pi}_{DL} + \tilde{\pi}_{DD}} \tag{42}$$

After calculating the packet loss rates according to the packetization mechanism used, we can now express the mean absolute difference (MAD) for frame N, for either Intra (I) or Inter (R) coded MBs as:

$$MAD(M_s^N) = \frac{\sum_{j=1}^{256} |f_{sj}^N - E[\hat{f}_{sj}^N]|}{256} \tag{43}$$

The next step is the calculation of the expected value of a reconstructed pixel at the receiver $E[\hat{f}_{sj}^n]$. This value, will be equal to the reconstructed value at the encoder times the probability to receive the MB correctly $\tilde{\pi}_R^{(i,n)} \tilde{f}_{ij}^n$, plus the probability to loose this MB and so use the reconstructed value of the same pixel of the previous frame $E[\hat{f}_{ij}^{n-1}]$ (error concealment is used). In the second case, if there is an MB before it, the decoder can use different EC method. This event requires the addition of the probability that this MB is lost and the previous MB (f_{ml}^{n-1}) was received correctly which will be the value of this pixel

in the previous frame ($E[\hat{f}_{ml}^{n-1}]$). Finally, we have to add the probability that both the previous and the current MB are lost and another pixel from the previous frame ($E[\hat{f}_{ij}^{n-1}]$) is used. The value $\tilde{e}_{ij}^{n}$ in the following equations, denotes the IDCT residue that is added to the reconstructed pixel values. For intra encoded frames we have:

$$E[\hat{f}_{ij}^{n}] = \begin{cases} \tilde{\pi}_{R}^{(i,n)} \tilde{f}_{ij}^{n} + (\tilde{\pi}_{L}^{(i,n)} + \tilde{\pi}_{D}^{(i,n)}) E[\hat{f}_{ij}^{n-1}] \text{ if } M_{ij} \text{ is first MB} \\ \tilde{\pi}_{R}^{(i,n)} \tilde{f}_{ij}^{n} + (\hat{\pi}_{RL}^{(i,n)} + \hat{\pi}_{RD}^{(i,n)}) E[\hat{f}_{ml}^{n-1}] \\ +(\tilde{\pi}_{LL}^{(i,n)} + \tilde{\pi}_{LD}^{(i,n)} + \tilde{\pi}_{DL}^{(i,n)} + \tilde{\pi}_{DD}^{(i,n)}) E[\hat{f}_{ij}^{n-1}] \text{ otherwise} \end{cases} \tag{44}$$

For inter frames we obtain similarly:

$$E[\hat{f}_{ij}^{n}] = \begin{cases} \tilde{\pi}_{R}^{(i,n)} (\tilde{e}_{ij}^{n} + E[\hat{f}_{uv}^{n-1}]) \\ +(\tilde{\pi}_{L}^{(i,n)} + \tilde{\pi}_{D}^{(i,n)}) E[\hat{f}_{ij}^{n-1}] \text{ if } M_{ij} \text{ is first MB} \\ \tilde{\pi}_{R}^{(i,n)} (\tilde{e}_{ij}^{n} + E[\hat{f}_{uv}^{n-1}]) + (\tilde{\pi}_{RL}^{(i,n)} + \tilde{\pi}_{RD}^{(i,n)}) E[\hat{f}_{ml}^{n-1}] \\ +(\tilde{\pi}_{LL}^{(i,n)} + \tilde{\pi}_{LD}^{(i,n)} + \tilde{\pi}_{DL}^{(i,n)} + \tilde{\pi}_{DD}^{(i,n)}) E[\hat{f}_{ij}^{n-1}] \text{ otherwise} \end{cases} \tag{45}$$

4.4.1 Rate Distortion Optimized Mode Selection (RDOMS)

The next question that logically comes up is how can the previously developed models can be utilized for end-to-end video streaming in such a manner that parameters like the expected distortion is minimized. The answer to this question requires an online solution to a multi-variable optimization problem. However, the closed form equations that we developed assure that an implementation of them can be realistic and provide real benefits when selecting the encoding mode of each specific macroblock.

Given that we have an estimate of MAD, i.e. the distortion at the decoder, by taking into account all the necessary parameters like the packetizer, network channel status, TCP introduced latency, and error concealment method, the encoder can apply RD optimized mode selection for each macroblock. Consider a group of Φ macroblocks that belong to frame n, with $\Phi_g^n = (\Phi_g^n, ..., \Phi_{g+N_G-1}^n)$ and N_G the number of MBs in unit Φ. Assuming that there is a number of macroblock groups that belong to F frames, the objective is to:

$$\min(E[D(f^F, \Phi_g^n)]) \quad \text{such that} \quad R(f^F, \Phi_g^n) \leq R_c \tag{46}$$

with the expected distortion for macroblock $\Phi_g^n \in f^n$:

$$MAD(f^n, \Phi_g^n) = \frac{\sum_{j=1}^{256} |f_{sj}^n - E[\hat{f}_{sj}^n]|}{256} \tag{47}$$

The Lagrangian optimization problem is therefore defined as:

$$\min \sum_{j=1}^{F} \sum_{i=1}^{N_G} J_k = \sum_{j=1}^{F} \sum_{i=1}^{N_G} E[D(f^j, \Phi_{g+i}^n, type)] + \xi \sum_{i=0}^{N_G} R \tag{48}$$

where *type* is inter or intra type of macroblock. The selection of the optimal Lagrange multiplier ξ, can be selected using several alternatives. In order simplify the algorithm we selected as as in [112], in order to simplify comparison. So for frame n this parameter is set as:

$$\xi_n = \frac{2B_n + (\beta - B_n)}{B_n + (\beta - B_n)} \xi_{n-1} \tag{49}$$

where B_n is the buffer occupancy after the encoded frame n, has been added to the buffer.

4.4.2 Experiments

The network testbed presented in figure 20, was used throughout our experiments in this chapter. Both the sender and the receiver were linux boxes while the middlebox was a freeBSD machine that acted as a router. The Dummynet software [36] was used in the middlebox in order to emulate various link configurations in terms of packet loss rate, bandwidth and delay. The QCIF FOREMAN and AKIYO sequences [94], were used for real-time encoding with the H.263 encoder at various bitrates for the streaming experiments [44]. The video packets were packetized into RTP packets and then sent to TCP. Due to the short duration of the sequences (150 frames), they were repeated and fed as input to the encoder. The capacity of the bottleneck link between the two routers is set to 250Kbps. The results were obtained by running the same scenario 100 times and averaging the PSNR values of the same experiments.

Figure 21 presents PSNR as a function of the channel packet loss probability for a target real-time encoding rate of 256Kbps and 64Kbps respectively. We compare our approach with results reproduced from [112], where the authors proposed an RD optimal mode selection (RDOMS) algorithm for streaming with UDP. We also evaluate the approach reported

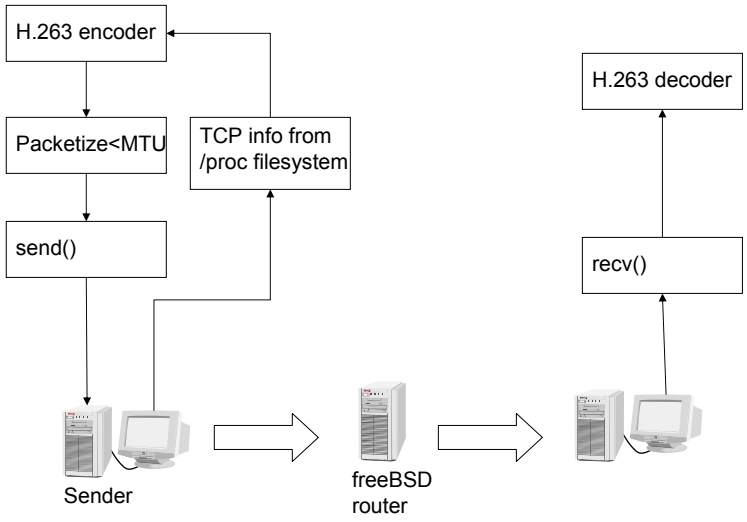


Figure 20: Experimental setup for real-time video streaming.

at [72], that also considers streaming with TCP. We see that when the target bitrate was 256Kbps, the RDOMS/UDP approach outperforms both the other two. However, the benefit of the proposed RDOMS algorithm, comes into play when TCP is used for transport. It clearly outperforms by 2-2.5 db, a purely TCP based streaming approach, which lacks an "understanding" of the network. Most important, for higher packet loss rate, the performance in terms of PSNR is increasing. When the target bitrate was set to 64Kbps, PSNR presents the same trend, but this time the effect is not so severe, due to the lower bitrate injected into the network.

4.5 Playback Buffering and Transport Protocol Performance Models

Having modeled the expected decoder distortion, we now proceed to derive a model that couples the behavior of the TCP transport protocol with the playback buffer at the receiver. Playback buffering at the receiver is usually employed in order to accommodate network

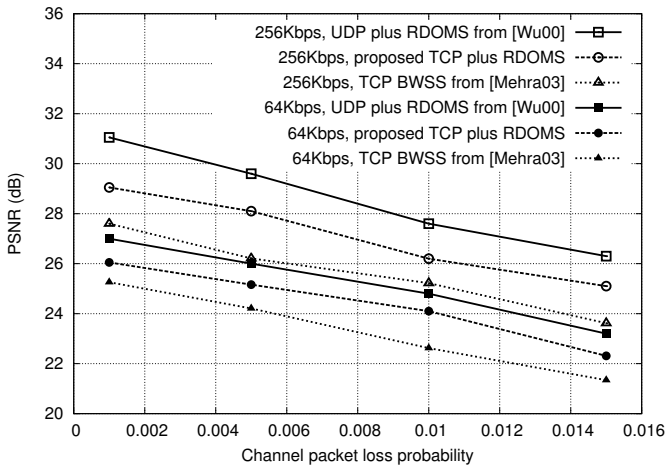


Figure 21: PSNR as a function of the end-to-end packet loss probability for video streaming with TCP.

delay fluctuations (i.e. jitter). Apart from network induced delay fluctuations, transport protocols also affect jitter since they control packet retransmissions. Therefore, a proper model of the network and the transport protocol has to be used in order to accurately quantify the effects of jitter. The lack of proper modeling is attributed mainly to the use of UDP as the transport protocol. However, in order to correlate the behavior of specific

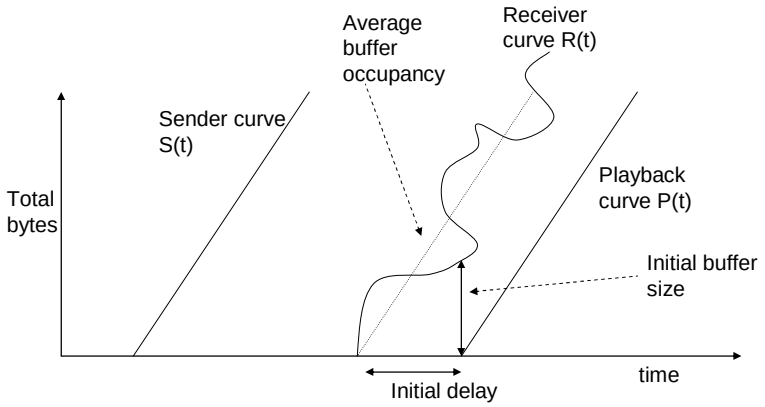


Figure 22: Sender, receiver, and playback buffer curves.

transport protocol at the receiver with the playback buffer and the expected distortion, we

have to derive a model for the protocol. Because this problem is orthogonal to the global model we want to derive, we provide the model that characterizes TCP goodput at the receiver in the appendix A. Therefore, with the receiver model, we will be able to derive a better approximate of the receiver function $R(t)$. As a next step we would like to define the optimal playback curve $P(t)$ (figure 22). Concerning TCP's goodput we have that:

$$E[G] = \frac{E[R] + P_{TO}E[R_{TO}]}{(E[X]+1)RTT + E[Z^{TO}]P_{TO}} \tag{50}$$

and we also showed in the appendix, that the goodput variance is:

$$E[V] = \frac{\frac{8}{pb} + (\frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \left(\frac{2+b}{3b}\right)^2})(1/2 - 4/p) + 1/p^2}{(E[X]+1)RTT} \tag{51}$$

As a next step we would like to derive bounds concerning the expected number of received bytes. By using Chebychev's inequality [89] we have:

$$P(|R - \mu| \geq \epsilon) \leq \frac{Var(R)}{\epsilon} \tag{52}$$

This equation means that the probability that the number of packets received is between these two bounds will be higher than $1 - \frac{V(R)}{\epsilon}$. We use the notation $lb(t)$ and $ub(t)$ to signify the lower and upper probability bounds respectively .

Based on the derived probability bounds, there are two parameters that have to be calculated for the playback buffer, and these are the minimum initial playback delay Δ, and minimum playback buffer size B. Given that that the playback curve is symbolized as $P(t)$, then the initial playback delay should be selected as:

$$\Delta \geq max(t - P^{-1}(lb(t))) \tag{53}$$

In this equation, P^{-1} expresses the pseudo-inverse function of the playback curve at the receiver [102]. Concerning the initial buffer size it should be selected as:

$$\begin{aligned} B &\geq max(ub(t) - P(t-\Delta)) \Rightarrow \\ B &\geq max(E[R] + Var(R) - P(t-\Delta)) \end{aligned} \tag{54}$$

The most important problem that can occur at the receiver during the playback of a streamed video sequence is a buffer underflow. For this to happen, the latency variance of the packets in flight must be more than the initial playback delay Δ. Figure 22 depicts this situation very clearly by showing the several curves that are involved in the playback process. This constraint can analytically be expressed in the following form given the previously calculated quantities: $R(t) \geq P(t) \Rightarrow G(R)t \geq c_1(t - \Delta)$. Given that the playback rate is c_1 bytes/sec, the average buffer occupancy at the receiver will be:

$$E[B] = \frac{E[R] + E[V]}{(E[X] + 1)RTT} - c_1((E[X] + 1)RTT) \tag{55}$$

If we solve equation 55 for c_1 we get the optimal playback rate given specific network conditions. Moreover, the probability P_D that was estimated earlier, described the probability for a packet to be delayed due to retransmissions or due to network delayed packets that are also invalid for playback. So for TCP this value could also be calculated as follows, by revisiting equation 34:

$$\begin{aligned} P_D &= P\{\text{latency} + \text{time sent} > \text{deadline}\} \\ &\text{or} \\ P_D &= P\{\frac{ds}{G} + t_s > t_d\} \end{aligned} \tag{56}$$

where t_s and t_d denote a packet's sent time and its playback deadline respectively. Now the expected reception time for a data chunk with size ds will be:

$$E[t_r] = E[t_s] + \frac{ds}{E[G]} \tag{57}$$

If we account for the playback delay, the probability of successful playback will be:

$$P_{success} = P\{\frac{ds}{G} + \frac{b_1}{c_1} + t_s < t_d\} \tag{58}$$

By solving the above equation, we can obtain the optimal playback rate c_1, for a given buffer occupancy b_1.

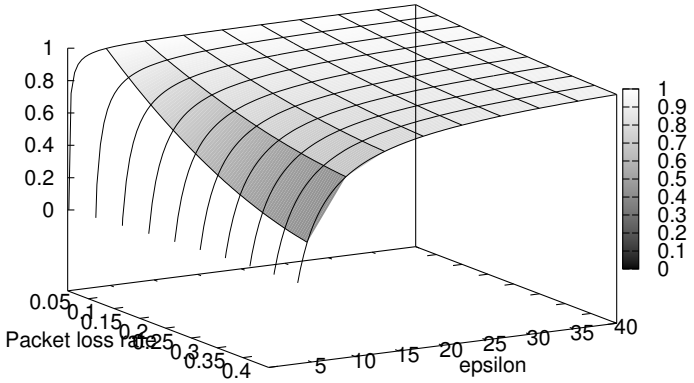
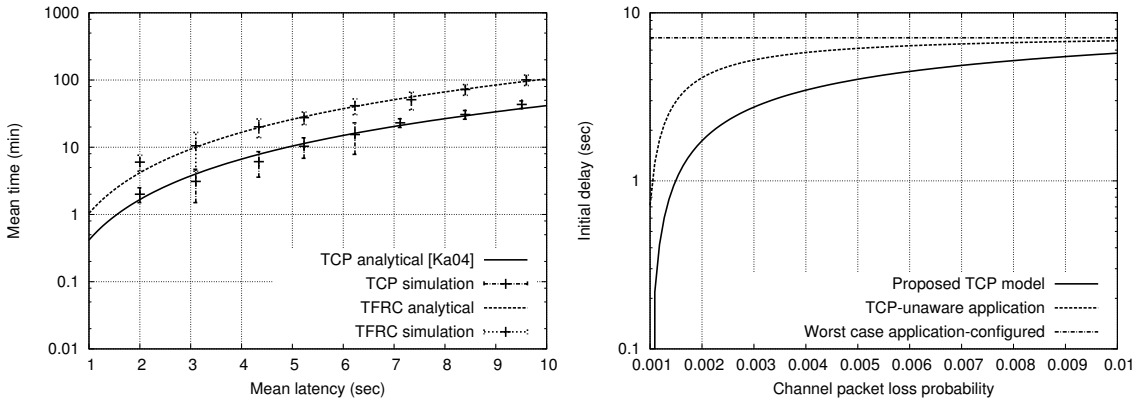


Figure 23: Probability for the buffer not underflowing for varying packet loss rate and the allowed packet delivery bound with TCP.



(a) Mean time between underflow events

(b) Initial preroll delay (Δ) needed to achieve a BUP<0.01

Figure 24: Numerical and simulation results for validating the playback buffer model. Parameters: $RTO_0 = 200$ ms, $MSS = 1460$ bytes, $W_0 = 1$ segment, $W_{max} = 6$ MB, video duration of 100 sec.

4.5.1 Experiments

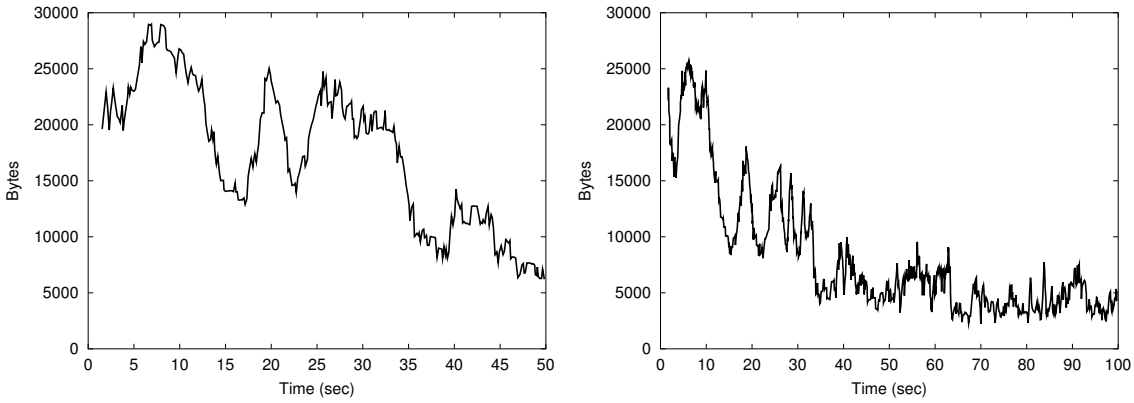
Playback Buffer Performance: Experimental results compare the proposed models with similar work identified in the literature. We reproduce the experiment reported at [102], where the authors lack a complete end-to-end model for the protocol being used, which in their case was UDP.

Figure 23 presents the probability of the playback buffer not underflowing as a function of the packet loss rate and the packet delivery bound when TCP is the transport protocol. The important observation from this figure is that despite the increase in packet loss rate, the desired packet delivery bound is the one that dominates the probability of the buffer being full (BFP). Being more elastic with the bound value, the receiver can achieve higher BFP but of course with lower throughput, when packet loss probability is increased. A very interesting result is presented in figure 60(a). It presents the mean time between two buffer underflow events as a function of the end-to-end delay or latency between the two endpoints. When the TFRC protocol was used, it produced smaller number of buffer underflow events than TCP as a result of its stable output rate.

Figure 60(b) depicts the initial delay needed at the playback buffer, in order to achieve a buffer underflow probability (BUP) less than 1%, as a function of the packet loss rate for TCP. It is obvious that the value of the initial delay is increasing very fast as the packet loss probability is increased until it saturates in a value close to 7 seconds for a video stream that has duration of 100 seconds and is making estimation using the proposed TCP model. However, a streaming application, that is using TCP and attempts to simply estimate the network induced latency results into a high rate of buffer undeflow events early in the video streaming process. In order to overcome these underflow events, a larger initial delay Δ is selected, by using estimate of the RTT [104]. The line called "worst case", represents a fixed value for Δ that an "ideal" application could select if it had knowledge of the packet loss probability range.

Adaptive Playback Buffer: An adaptive playback buffer is capable of accommodating delay fluctuations for the received packets. With an adaptive playback algorithm, in the short-run the playback frame rate is kept relatively constant, while it may change significantly over time depending on network conditions. This type of playback buffer algorithm is well suited for a reliable transport service, like the proposed system which is based on TCP, since it can "conceal" the delay variations.

We tested a playback adaptation according to the derived equation 56 that expresses our comprehensive model. For this experiment we added TCP cross traffic of 2 FTP flows.



(a) Buffer size evolution over time at the receiver for a UDP-based system.

(b) Buffer size evolution over time for a TCP-based system with the proposed playback adaption strategy.

Figure 25: Buffer size evolution at the receiver with cross traffic of three FTP/TCP flows.

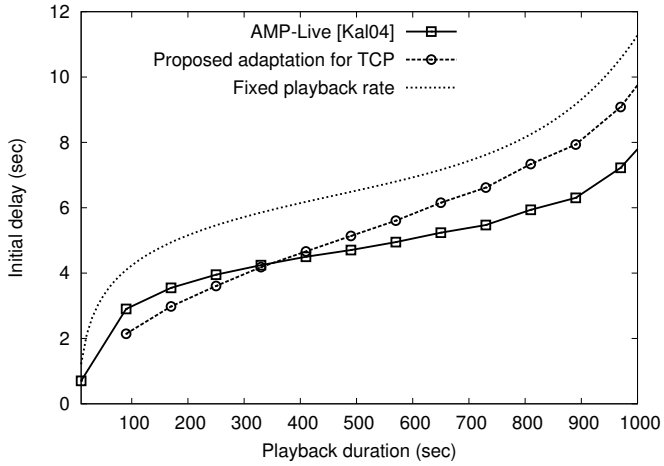


Figure 26: Experimental results of the required initial delay Δ needed to achieve a BUP$<$0.01.

Results for the buffer size versus time are shown in figure 25. Ideally the UDP-based system would maintain a relatively fixed varying range for the buffer size. Nevertheless, the FTP/TCP cross-traffic results into congestion for some periods and this further creates UDP packet losses. That is why we observe the playback buffer drain for some periods. However, for the case of the proposed playback speed adaptation (equation 56) for TCP, we can see that for the first seven seconds (figure 25(b)), the buffer occupancy is relatively high and in addition data are removed with low frequency. This means of course a low frame rate at the receiver. However, when congestion incident has passed (around 10 seconds),

the end-to-end delay is reduced and proposed playback adaptation, is capable of delivering more frames. In addition, the playback algorithm adapts to the lower delay, and provides a higher frame rate and reduced buffer requirements. This results, are very important, since they depict the importance of having a good adaptive playback algorithm when a reliable video streaming system is employed.

Figure 26 compares the playback buffering performance models between the proposed joint transport protocol and playback buffering model and results from [55]. More specifically, we depict results for the initial required delay Δ, before the playback commences, versus the duration of the video sequence. At [55], the authors als defined a adaptive playback adaptation strategy, that is based on a model that considers single two-state Markov channel, and a deadline constraint ARQ mechanism. However, with the more accurate path model that we adopt, we get a better estimate of the end-to-end latency when a heterogeneous path with wireless and wired links exist. This is the reason why our system needs a shorter strtup delay as figure 26 indicates.

4.6 RD Optimized Packet Scheduling

In section 4.4, we presented detailed model of the expected decoder distortion when TCP is used as the transport protocol. In this section we extend the problem of optimal video streaming one step further: Given the encoder buffer which contains a set of macroblocks Φ_g^n, that belong to frame f^n (for F possible frames), how can we define the optimal transmission schedule s for TCP, so that $E[D(f^F, \Phi_g^n)]$ is minimized? How can we take into account the TCP induced latency and its effect on buffer occupancy, as calculated in section 4.5? Therefore for a given set of allowed schedules S, the first of the previous two questions can be formally written as:

$$s = arg \min_{s \in S} E[D(f^F, \Phi_g^n)] \tag{59}$$

The search space of the possible schedules S, depends on the number of macroblocks in the buffer. Clearly an exhaustive search does not represent a practical solution. However, in [73] the authors considered a similar problem for layered encoded video, and they devised

a simple greedy search algorithm for finding schedules that represent suboptimal local solutions. Main feature of the greedy algorithm is that it is less ambitious than a more thorough algorithm in the sense that it continuously looks for a better video packet, in terms of distortion, rather than the best video packet. We implemented a simple algorithm for greedy search, and we found that the actual overhead is no important for real-time encoding since the number of encoded macroblocks and frames that reside in the encoder buffer is usually small. The suboptimal greedy search problem is defined for sets of MBs that belong to the same frame, or in the case where the number of frames in the encoder buffer is small (less than 3), then the number of MBs in a single GOP defines the size of each subproblem.

Note that the additional playback constraint (section 4.5) essentially means that not only we should transmit the data packet that will minimize the expected distortion, but also the packet that is actually expected to arrive at the client by the deadline. This is what lines 4-9 of the algorithm in figure 27, are exactly doing by identifying the set of MBs that will actually reach the decoder in time ($\Phi^{\#}$). After this decision is made, the algorithm invokes the greedy scheduling algorithm which identifies the optimal schedule (lines 11-14). The relative priority of each MB is enforced through a prioritization mechanism so that MBs that affect distortion the most are committed to TCP first. We also depict in the last line of the algorithm, a statement that indicates the clients ability to provide feedback to the sender. The feedback may involve among others, the actual receive time for a packet and the actual goodput, allowing thus the implementation of more efficient estimators at the sender.

4.6.1 Experiments

Figure 28 presents PSNR as a function of the channel packet loss rate. We compare the packet scheduling algorithm with the previously developed schemes that do not employ any intelligence in the scheduling process, namely TCP-RDOMS and vanilla TCP streaming. For smaller packet loss rate, packet scheduling can have an effect on the decoder PSNR since a small number of packets are lost. However, as the packet loss rate is increased, the use of an optimal packet schedule has less effect since there is higher probability for the

steady_state()

1: $G \leftarrow$ // estimate TCP goodput, equation 50
2: $\tilde{r}_i \leftarrow$ // estimate arrival for MB_i^n, equation 57
3: $\tilde{P}_D \leftarrow$ // estimate P_D, equation 56
4: **for all** MB $i \in \Phi$ **do**
5: **if** $\tilde{t}_{r_i} < t_{d_i}$ **then**
6: Add MB_i in $\Phi^{\#}$
7: **else**
8: Drop MB_i
9: **end if**
10: **end for**
11: **for all** $MBs \in \Phi^{\#}$ **do**
12: Find optimal set Φ^* that min(E[D]) // equation 119
13: $set\_priority(send\_buffer, MB_i)$
14: **end for**
15: send(MBs$\in \Phi^*$)

recv_report()

1: Client may report in feedback messages the both $\tilde{e} = |t_r - \tilde{t}_r|$ and G.

Figure 27: Pseudo algorithm for two-stage optimal schedule construction at the streaming server.

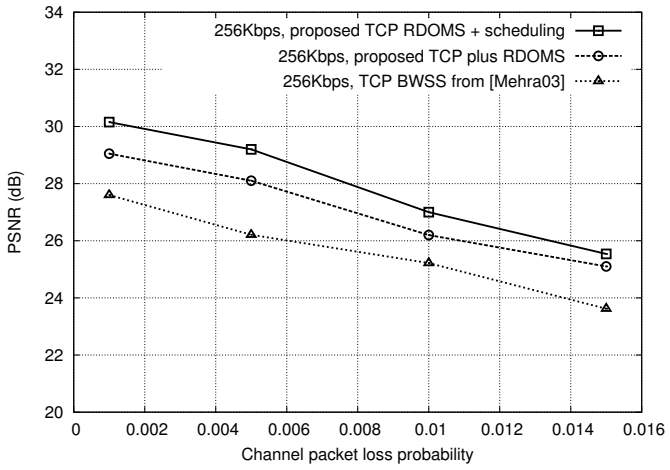


Figure 28: Results for streaming experiment with the proposed RD optimized packet scheduling algortihm.

packet to be lost. Nevertheless, on average a gain of nearly one db can be obtained for this scenario. We plan to explore the tradeoff when different MB set sizes are selected for the scheduling algorithm, over the introduced latency. However, the search space for real-time video streaming is limited since the number of encoded frames has to remain small in order to avoid excessive delays.

(a) TCP

(b) Proposed RD optimized packet scheduling with TCP

Figure 29: Captured frames for the proposed RD optimized packet scheduling algorithm.

Frames in figure 29 were captured for two different RD optimized packet scheduling algorithms. In figure 29(a) we present results obtained from TCP-based streaming. The main visual impairment that we observe is the distortion in a specific area of the received frame. The problem that caused this behavior is the burst loss, which made impossible even for error concealment to produce a visual result. However, the proposed system 29(b) avoids the previous behavior for the following reason: It avoids scheduling critical packets at the same instant since it estimates the increased packet loss that the packet will suffer.

4.7 Conclusions

In this chapter we presented an analytical study that characterizes the performance of video streaming with the transmission control protocol (TCP). Initially, we developed an analytical model of the expected video distortion at the decoder with respect to the TCP parameters, channel state, and error concealment method at the receiver. Based on this model we proposed an algorithm for RD optimized mode selection (RDOMS) for video streaming with TCP. Experimental results for real-time video streaming showed PSNR improvement of nearly two db over currently proposed TCP-based streaming mechanisms. The next contribution is the development of a joint model of the TCP protocol, and the playback buffer at the receiver. Based on this model, we derived the optimal playback rate

at the decoder. Subsequently, based on the two models, we proposed an algorithm, for RD optimized packet scheduling with TCP. Our results show an additional improvement of nearly one db, when packet scheduling is applied together with the RDOMS algorithm.

Therefore, we can see that TCP presents a viable solution for video streaming applications. Moreover, we showed that if additional optimizations can be preformed at the video encoder side, further quality improvement can be observed. The wide-scale deployment of TCP, and the ability to realize the proposed algorithms at the application level, can assure that the proposed streaming mechanism presents a practical video streaming solution.

CHAPTER V

MODELING THE EFFECT OF HANDOFFS ON TRANSPORT PROTOCOL PERFORMANCE

In this chapter, we present a new analytical model for studying the effects of mobile handoffs on the performance of transport protocols. Specifically, we develop analytical models that characterize the throughput, latency, and jitter of TCP and TFRC as a function of the wireless handoff induced packet loss rate and the relative disruption time. The result of our analysis is a modular performance evaluation model, that may be used for analyzing the effect of various mobility management architectures and mobility scenarios for existing and emerging transport protocols. The development of this model for the case of bulk traffic workload, is essential for analyzing and modeling the performance of media applications in the next chapter of this dissertation.

5.1 Introduction

The wide-spread success of IP-based mobile and wireless devices, has created a need for new network protocols and architectures so that revenue-generating and seamless multimedia services can be provided to the end user. One of the first challenges that has to be resolved is that of mobility management. The functionality that mobility management defines consists of two separate operations — location management and handoff management. Currently, the protocol that is considered to offer a practical solution to the above problems is mobile IP [83], since it offers a solution to these two problems at the same time. First, it allows a host to be reached through a static IP address (location management), which is called the home address (HoA). Second, it allows transport layer sessions like TCP connections to continue when the underlying host moves (in mobile applications) and changes its IP address (handoff management). The latter is important to higher layer protocols that offer reliable service delivery [103, 88].

In general, TCP performance suffers from several problems due to handoffs caused by host mobility: The first problem is related to blackouts that lead to successive timer expirations and increase of the RTO every time an unsuccessful transmission takes place [99, 10]. A mechanism for partly resolving this problem is through explicit layer 2 notifications to TCP, so that it can freeze the RTO [47, 75]. The second problem is the long and fluctuating delays due to local retransmissions in the wireless link [26, 28] or due to deep buffering in cellular access networks [28]. This situation can result into the invocation of congestion control algorithm and substantial decrease in the throughput. Third, packet losses due to wireless errors that have a similar effect since TCP is unable to distinguish them from congestion-induced losses.

Several methods have been proposed to resolve these problems caused to TCP [103, 9, 10]. Some mechanisms are related to split connection protocols [10], and others to mechanisms for end-to-end differentiation of congestion and wireless losses based on delay measurements [24]. While TCP-compatible rate control protocols have been studied extensively (e.g., the IETF standardized TCP-friendly rate control protocol (TFRC) [46]), there has not been much study of for their behavior during wireless handoffs. We are aware of only one recent study, reported in [42], where the authors evaluate the performance of TFRC during handoffs between asymmetric networks. The authors identify performance degradation in TFRC due to its slow-responsiveness, and suggest the use of L2 triggers so that the protocol can adjust faster to the conditions of the new link. However, there is no detailed performance analysis that incorporates the used mobility protocol.

There are several models for quantitatively analyzing TCP throughput [80, 95], and latency[22], specifically in the context of the wired internet. Considerable amount of work on modeling mobility management protocols is available, and especially for mobile IP and its derivatives [31, 4, 114, 15, 16]. The main focus of the these mobility management modeling approaches has been the characterization of the signalling and processing loads as a measure of protocol performance [114, 71, 79]. Some related recent efforts include [16], where the authors investigate the performance of two IETF handoff protocols, namely pre/post-registration for UDP. They develop a simple analytical model that estimates the packet loss

rate and delay. Their results are restricted to the case of CBR unregulated traffic sources. Similar approaches, that do not consider complex transport protocols, and instead focus on UDP, have been published for mobile IP and mobile IP with route optimization [15, 31].

5.2 Transport Protocol Performance Models

We now propose a model that attempts to capture the behavior of TCP and TFRC in during handoff between heterogeneous networks.

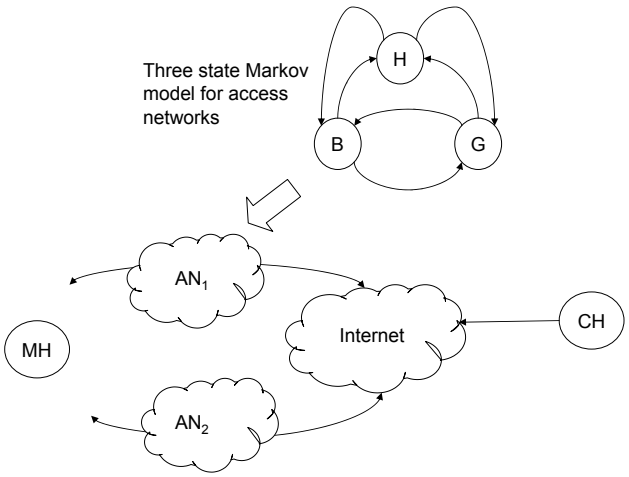


Figure 30: End-to-end path model for transport protocol characterization during handoffs.

5.2.1 Heterogeneous Network Model

Consider an end-to-end session with the data flowing from the correspondent host (CH) to the mobile host (MH) as figure 30 indicates. According to the mobility scenario, the MH is initially associated with the first access network (AN_1), and at some time in the future it may move towards AN_2. The two access networks are modeled separately, while they are both attached to the core network (C) which is the Internet. We make the assumption that while the mobile host is connected to AN_1, the transport protocol is in steady state. Subsequently, after handoff is performed to AN_2, the protocol will reach the steady state at some time in the future.

The core network is modeled as a two-state Markov chain, an approach that has been shown to predict Internet packet loss behavior quite well [82]. The access networks are modeled as a three-state Markov chain, in which the three states are good (G), bad wireless (B) and handoff (H). This implies that we separate the handoff induced packet losses, from other packet losses due to other reasons in the AN. Note that the adoption of a different path model, will not alter the analytical formulas developed in this chapter, but may affect the accuracy of the derived packet loss probability.

Note that the bad state for an AN corresponds to bad state due to a handoff. The benefit of creating different models for the different access networks, is that the characteristics of heterogeneous networks can be captured when handoff takes place. We also have to mention, that each end-to-end path, that corresponds to one of the two access networks and the core network, is characterized by different forward and backward trip times (FFT and BTT). This path model is general enough so that it can accommodate the modeling of different handoff management protocols. In the appendix B, we derive simple models for hierarchical mobile IP (HMIP), and mobile IP with route optimization (MIP-RO) that can inter-operate with our path model. From these simple models, we derive two values: the packet loss probability p_h, and the disruption time X for each mobility management protocol. For the rest of the analysis in this section, we assume that the MH is using Mobile IP so that the TCP connection does not break due to a change in the IP address.

5.2.2 Handoff Scenario and TCP Behavior

We model a TCP connection between two endpoints by considering rounds (as in [62]), that have a duration of one RTT. We name the number of RTT rounds that pass until there is a packet loss as the NL round (figure 31). During this round, TCP sends a burst of packets equal to the allowed window, and waits for acknowledgments. This approach, is based on the renewal theory properties, similar to the methodology we developed in chapter 3. We selected this method so that we can obtain a closed form solution that can be used for practical applications.

We now describe the behavior of TCP during a handoff of duration X when mobile IP is

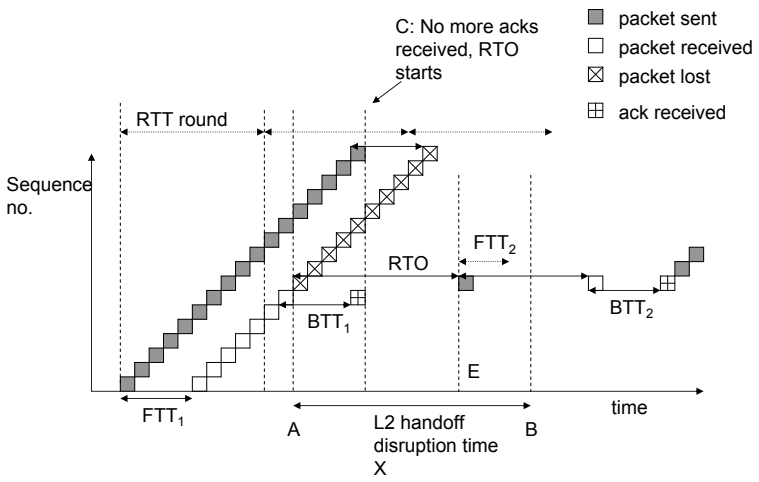


Figure 31: Packet-level TCP behavior during IP layer handoff.

used. Figure 31 depicts the packet-level behavior of both the TCP sender and receiver. We define t_A and t_B as the time instants at which handoff was initiated and ended ($X = t_B - t_A$). We assume that TCP is in steady state and sends packets to the MH that is attached to AN_1. As the MH moves, at some time instant, it will break its L2 connection with AN_1, and several packets can be lost as figure 31 indicates. After this happens, the sender will be receiving acknowledgments that are in flight and were sent before the MH broke connections to AN_1. These ACKs will trigger the sending of more packets. When the flow of ACKs stops (at time instant t_C), the sender will not be able to clock out new packets. Depending on X and p_h, the RTO may keep increasing until connection has been established or the session times out. In the next subsection, we will derive the value for p_h.

5.2.3 Handoff Induced Packet Loss

The probability of handoff induced packet losses is the probability of the one way latency L, being smaller than the handoff duration X:

$$p_h = P\{L < X\} \tag{60}$$

We define as f_L the distribution of the end-to-end latency. Concerning the disruption time X, it is a parameter that depends on the mobility management protocol used. In general it can have a fixed value for a specific protocol [114]. Therefore, the only random variable that has to be calculated in equation 60, is that of the network latency L or FTT. We model the one way network latency distribution f_{L_N}, as a shifted Gamma distribution with probability density function [25, 64]:

$$f_{L_N}(t) = \begin{cases} \frac{\lambda e^{-\lambda t}(\lambda t)^{\nu-1}}{(\nu-1)!} & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}$$

and a cumulative distribution function (c.d.f.) of:

$$F_{L_N}(t) = \begin{cases} 1 - e^{-\lambda t}\frac{(\lambda t)^{\nu}}{\nu!} & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases}$$

with mean ν/λ, and variance ν/λ^2. Therefore, by using the previous equations, we can obtain the packet loss probability due to handoff p_h:

$$p_h = \frac{e^{-\lambda X}\lambda^{\nu}}{\nu!}\left(e^{-\lambda RTT}(X + RTT)^{\nu} - (X)^{\nu}\right) \tag{61}$$

Handoff with Forwarding Buffer: We now account for the case where a mobility management protocol employs a buffering mechanism at the old AN, when a MH performs a handoff to the new AN, so that in flight packets are not lost. If we assume that the server where the buffering is performed, is modelled as an M/M/1/C queue [16], with finite capacity C and average load ρ, then the packet loss probability due to handoff has to be recalculated. The packet loss probability now is equal to the conditional probability that a packet arrives faster in the access network, given that the buffer is full:

$$p_{hb} = P\{L < X | N = C\} \tag{62}$$

So the conditional probability mass function will be $F_{hb} = P\{L < X \text{ and } N < C\}$, given that F_N is the c.d.f. of the M/M/1/C queue. Eventually, this will give:

$$\begin{aligned} p_{hb} &= F_{L_N}(X)F_N(C) \\ &= p_h\frac{\rho^C(1-\rho)}{1-\rho^{C+1}} \end{aligned} \tag{63}$$

Therefore, equations 61 and 63, describe in a parametric form, packet loss probability directly attributed to handoff.

5.2.4 TCP Throughput

Having calculated the packet loss probability due to handoff p_h, we now calculate the actual number of packets lost. This number, will depend primarily on the the number of in-flight packets which is the value of the congestion window. Since TCP is a window-based protocol [99], assume that the sender has sent a window w worth of packets in an RTT round. The probability that the first k packets are acknowledged in this round, given that the rest is lost because of a handoff or packet loss in the wireline path, is given by: $P_{H1}(w,k) = P_H(w,k) \bigcup P_1(w,k)$. Because packet loss in the wired path and handoff loss are independent, the previous equation becomes:

$$P_{H1}(w,k) = \frac{(1-p_h)^k p_h}{1-(1-p_h)^w} + \frac{(1-p_1)^k p_1}{1-(1-p_1)^w} \tag{64}$$

since the cause of the (k+1)-th packet loss while the MH is attached to AN_1 can either be due to handoff p_h, or due to packet loss in the wired link of AN_1 (p_1).

Case 1, $t_B < t_C + FTT_1$: From figure 31 we can see that if $t_B > t_C + FTT_1$, then clearly no duplicate ACKs will be received at the sender, and the only way for TCP to resume the data flow is by expiration of the RTO of the first lost packet. On the other hand, as X shrinks, and if $t_B \leq t_C + FTT_1$ the probability to receive a number of the last packets (close to point C) is increased. We need to find the probability that at least three of these last sent packets, are received resulting into the generation of three duplicate ACKs. If this happens, then the sender would fast retransmit the first lost packet, resulting into a faster recovery. We can rewrite the previous equation $t_B \leq t_C + FTT\_L_1 \Rightarrow X \leq RTT_1$.

Therefore, the probability to loose m packets from the n sent in the handoff round is given by:

$$G(n,m) \quad = \quad p_h^m(1-p_h) \text{ if } m \leq n-1 \tag{65}$$

and the probability to receive less than three from a round of k send, leading thus to a TO is:

$$g(k) = \sum_{k=0}^{\infty} G(2+k,k)P[w=2+k] \tag{66}$$

where $P[w=2+k]$ gives the probability that the current window is $2+k$ packets. From equations 64 and 65 we can get the probability that a loss in a window of w packets will lead to a timeout. This value will be one if $w \leq 3$, since if one of the three packets is lost, not enough duplicate ACKs will be received.

If $w > 3$, the probability to have a TO will depend on an additional third component, besides the two that we just described:

$$P_{TO} = \sum_{k=0}^{2} P_H(w,k)(1+g(k)) + \sum_{k=3}^{w} F(w,k)h(k) \tag{67}$$

In this equation, the third component $F(w,k)$ represents the probability of a packet loss in a regular end-to-end path characterized by a packet loss rate p similar to equation 64.

Case 2, $t_B > t_C + FTT_1$: In the case where $t_B > t_C + FTT_1$, the retransmitted packet is lost and the value of RTO will increase as shown in figure 31. However, if the handoff disruption continues, the RTO could grow even larger. Therefore, the average duration of a timeout has to be derived, based on the probability that $t_B > t_C + FTT_1$. The number of the retransmitted packets that will be lost leading to further RTO increase determine when the data flow will be resumed on the new link. Given that the duration of a handoff is $X = t_B - t_A$, the number of retransmitted packets depends on the duration of the timeout period, and subsequently on the number of exponential growths the TO timer experienced. We know that for TCP, k consecutive RTO events will have a duration [99]:

$$L_{hk} = \begin{cases} (2^k - 1)RTO_0 & \text{if } k < 6 \\ (63 + 64(k-6))RTO_0 & \text{if } k \geq 7 \end{cases} \tag{68}$$

where RTO_0 is the initial value of the retransmission timer. By inverting this expression, we obtain the number of RTO expirations:

$$k = \begin{cases} log_2(\frac{L_{hk}}{RTO_0} + 1) & \text{if } L_{hk} \leq (2^6 - 1)RTO_0 \\ \frac{L_{hk}}{RTO_0} + 5 & \text{otherwise} \end{cases} \tag{69}$$

So the number of experienced TOs will be obtained by taking the $\lceil\ \rceil$ of $\frac{L_k}{RTO_0}$. This value will give the number of RTO expirations and the number of retransmitted packets. The previous equation finally gives the expected number of packets sent during $t_C < t < t_B$:

$$E[S_h] = \begin{cases} log_2(\lceil \frac{X}{L_{hk}} \rceil + 1) & \text{if } X \leq (2^6 - 1)RTO_0 \\ \lceil \frac{X}{L_k} \rceil + 5 & \text{otherwise} \end{cases} \tag{70}$$

and the duration of the induced TOs is $E[N_h^{TO}] = E[L_h] + X$.

Concerning the evolution of the congestion window, we next derive the equations that describe its evolution and the duration of a round in RTTs X until a packet loss occurs. Following the notation from chapter 3

$$E[A] = \frac{b}{2}E[W] = \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} \tag{71}$$

Finally, by combining all the previous equations, the complete TCP throughput model that considers handoffs between asymmetric links is given by:

$$T_{HTCP} = \frac{\frac{1-p_1}{p_1} + \frac{1-p_2}{p_2} + E[W_1] + E[W_2] + \frac{1-P_{TO}}{1-p_1} + P_{TO}E[S_h]}{RTT(E[A]+1) + RTO_0\frac{(1-P_{TO})P_{TO}}{1-p_1} + P_{TO}(X + E[L])}$$

This equation essentially includes for every stage of the handoff, the expected number of packets sent, over the expected time needed to send these packets. In this way we are able to fully quantify the throughput of TCP during a handoff between asymmetric links. It is interesting to note that we have described the TCP throughput as a function of the disruption time X, which is essentially controlled by the mobility management scheme in use, and the handoff packet loss rate p_h.

5.2.5 TCP Latency and Jitter in Congestion Avoidance

After developing an analytical model for TCP's end-to-end throughput, we subsequently model the latency for the transfer of a specific chunk of data bytes d when the protocol is in the congestion avoidance phase. TCP in general suffers from latency fluctuations, due to the retransmissions and the queuing delays in the network. So the latency for the delivery of a single packet, depends on the cause of the retransmission. If it is a TO the latency will be equal to the duration of the TO, while if it is a TD, a delay of an RTT will be experienced. We showed that the probability for a packet to be recovered with fast retransmission is $1 - P_{TO}$ (equation 67). Therefore, the latency of a single retransmitted packet is $L = P_{TO}L_k + (1 - P_{TO})RTT$, that can be due to an RTO expiration $P_{TO}L_k$ or due to a fast retransmission $(1 - P_{TO})RTT$. The average latency that a lost packet experiences is:

$$E[L] = P_{TO}E[N^{TO}] + (1 - P_{TO})RTT \tag{72}$$

Now the average latency for a chuck of d data bytes, will be:

$$L_d = \frac{d}{T_{HTCP}} \tag{73}$$

Special attention is given to the value of latency fluctuations (i.e., jitter), which is defined in [92] as the difference between the time instants that two packets were sent and the time instants at which the two packets were received. We can write this statement for two packets i and j as follows: $R_i - R_j - (S_i - S_j) = R_i - S_i - (R_j - S_j) = E[L_i] - E[L_j]$, while the average jitter is equal to the delay variance: $\bar{D} = Var(L)$. This value is derived from the previous as:

$$Var(L_d) \quad = \quad E[(L_d - E[L_d])^2] = E[L_d^2] - E[L_d]^2 \tag{74}$$

The values for $E[L_d]$ and $E[L_d^2]$ are easily calculated from the previously developed equations.

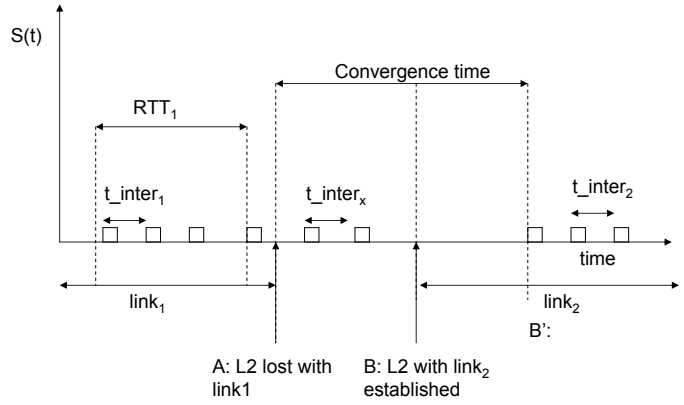


Figure 32: TFRC packet-level behavior during IP layer handoff.

5.2.6 TFRC Throughput and Latency

After dealing with TCP, we proceed with the characterization of TFRC. TFRC was introduced earlier as an equation-based rate control scheme. In this chapter we consider TFRC to be implemented on top of UDP, since UDP is protocol of choice for real-time media streaming applications. TFRC uses the closed form equation for TCP throughput in order to regulate the sender's output rate:

$$T_{TFRC} = \frac{s}{RTT\sqrt{\frac{2p}{3}} + RTO_0(3\sqrt{\frac{3p}{8}})p(1+32p^2)} \tag{75}$$

where s is the packet size, RTT is the RTT estimate, and RTO_0 is the value of the retransmission timer. In addition TFRC follows an inter-packet spacing algorithm at the sender:

$$t_{inter} = \frac{s\sqrt{RTT_{cur}}}{T * M} \tag{76}$$

M is the average of the square roots of the RTTs calculated using an explicit window moving average (EWMA), and RTT_0 is the most recent RTT sample [46]. Given that S_i and S_j are

the transmit times of two packets sent successively, the inter-packet spacing at the source is given as $S_i - S_j = t_{inter}$. However, the packet spacing at the receiver $R_i - R_j$, depends on the network queuing delays, which is the only cause of jitter in TFRC (figure 32). Note

```
1: if p > 0 then
2:    X_calc = T(p, RTT, T_0)
3:    tfrc_x = max[min(X_calc, 2 * X_recv), s/tmbi]
4: else
5:    if tnow − tld ≥ RTT then
6:       tfrc_x = max(min(2 * tfrc_x, 2 * X_recv), s/RTT)
7:    end if
8:    tld = tnow
9: end if
```

Figure 33: The TFRC rate estimation algorithm.

that, equation 75 does not represent the actual TFRC sending rate but only an upper bound for it. The actual output rate of TFRC is calculated using the algorithm in figure 4. In this algorithm, s represents the packet size, tld is time when the rate was last doubled, $tmbi$ is the maximum back-off time (64 seconds by default), and X_{recv} is the average receive rate. If p is zero, no packet loss has yet been seen by the flow and in this case, the TFRC sender emulates the slow start alforithm of TCP by doubling the transmission rate every RTT.

The main feature of TFRC is that it reacts slowly to RTT changes and this means that the throughput estimate changes slowly when compared with TCP. Therefore, a sudden change in the RTT and bandwidth of a link, as is the case with handoffs, will lead to considerable packet losses. We will now attempt to model this behavior. We start by identifying the RTT estimation procedure that TFRC uses. Given a decay factor df, the n-th RTT estimate is calculated as follows:

$$RTT_n \quad = \quad df * RTT_{n-1} + (1 - df) * \sqrt{RTT_{cur}} \tag{77}$$

with RTT_{cur} be the last actual RTT measurement. When the TFRC sender does not receive feedback during an entire RTT, it reduces the output rate by half. If we assume that at time instant t_B (figure 32), where handoff is over, the sender sends at least one packet and receives feedback, then after RTT_2 seconds, the sender will receive the first feedback

report. So for a handoff duration of X seconds, the sender will be gradually reducing the rate by half every RTT_1, since this was the last RTT estimate. Therefore the total number of packets sent during X will be gradually reduced with a total number sent during X:

$$E[S_h] = \sum_{i=0}^{\lceil \frac{X}{RTT_{cur}} \rceil} \frac{1}{2^i} T_{TFRC} \times RTT \tag{78}$$

Therefore, for two links with different characteristics, p_1, RTT_1 and p_2, RTT_2, the actual rate at the sender will be given for links 1,2 or when $t < t_A$ or $t > t_C$ (figure 32) by:

$$T_{TFRC}^{A^-,B^+} = \max\Big(2\min(T_{TFRC}, 2(1-P_{1,2})T_{TFRC}, \frac{s}{tmbi}\Big) \tag{79}$$

When $t_A < t < t_B$ then:

$$T_{TFRC}^{AB} = \frac{1}{X}\Big(\sum_{i=0}^{\lceil \frac{X}{RTT_{cur}} \rceil} \frac{1}{2^i} T_{TFRC} \times RTT\Big) \tag{80}$$

If $t_B < t < t_C$, the sender gradually starts to catch up with the new link bandwidth:

$$T_{TFRC}^{BC} = \frac{1}{t_{cv}}\Big(\sum_{n=1}^{n_{cv}} T_{TFRC}(RTT_n\_cv) \times RTT_n\Big) \tag{81}$$

We define as convergence time, t_{cv}, the time needed for TFRC to obtain its fair share of the bandwidth on the new link. Equation 77 can be expanded and written as:

$$\begin{aligned} RTT_n &= df^n * RTT_0 + df^{n-1}(1-df) * \sqrt{RTT_{cur}} \\ &+ \ ... + (1-df) * \sqrt{RTT_{cur}} \end{aligned}$$

which if we solve for n gives:

$$n_{cv} = log_{df}\Big(\frac{RTT_n - (1-df) * \sqrt{RTT_{cur}}}{RTT_0(1-df) - (1-df) * \sqrt{RTT_{cur}}}\Big) \tag{82}$$

Therefore, n_{cv} will give the number of RTT rounds needed in order for the RTT_n estimate to converge to the RTT_{cur}. In the handoff case, RTT_{cur} represents the RTT_2 of the new link while RTT_0 is the first estimate we had and it is RTT_1. Practically we would like for RTT_n to be close to 90% of the RTT_{cur}. So the total convergence time will be:

$$t_{cv} = X + n_{cv} * RTT_n \tag{83}$$

Concerning the expected latency for TFRC in congestion avoidance phase and it would be:

$$E[L_{TFRC}] = \frac{data\_size}{E[T^{A}_{TFRC}] + E[T^{AB}_{TFRC}] + E[T^{BC}_{TFRC}]} \tag{84}$$

5.3 Simulations

The WLAN topology that forms the basis of the simulations described in this section is shown in figure 34. The values for bandwidth and delay of the links are 1Mbps/100ms and 500Kbps/100ms for the GFA links, and two FA links, respectively. The case study simulated is now described — Initially, at time 5 seconds, the MH initiates an FTP data flow from the correspondent host (CH). According to the scenario, at time 50 the MH starts moving away from the first access point (AP), at a speed of 10m/sec, and is heading toward the other AP. The FA follows the Mobile IP procedure in order to notify the HA after the MH registers with new FA. In the case of Hierarchical MIP, the GFA is the one that handles the handoff from the two FAs that correspond to the two access points: AP_1, and AP_2. Finally, for the case of MIP-RO, we configured the MH to send a binding update directly to the CH. For all these experiments we used the ns-2 [77] network simulator with appropriate modifications when necessary. The simulations were run for 20 times, averaged results are shown in all the figures in this section.

5.3.1 Results for Mobile IP

By calculating the variance of the end-to-end latency, we essentially estimated the average value for the jitter. This will provide crucial insight into the precise correlation of transport protocol algorithms and the end-to-end jitter. The next step is to derive bounds on the average latency given a specific packet loss rate. Using Chebychev's inequality for the latency random variable L, we get:

$$P(|L - E[L]| \geq \epsilon) \leq \frac{Var(L)}{\epsilon} \tag{85}$$

The derived latency bounds for TCP are shown figure 35(a) and show that given a short disruption time, the latency is not sharply increased. This indicates that even if handoff

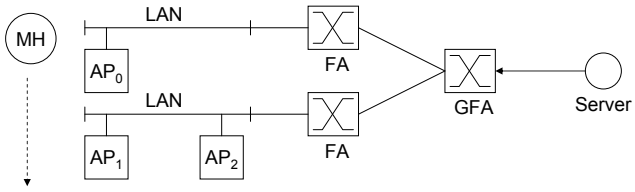
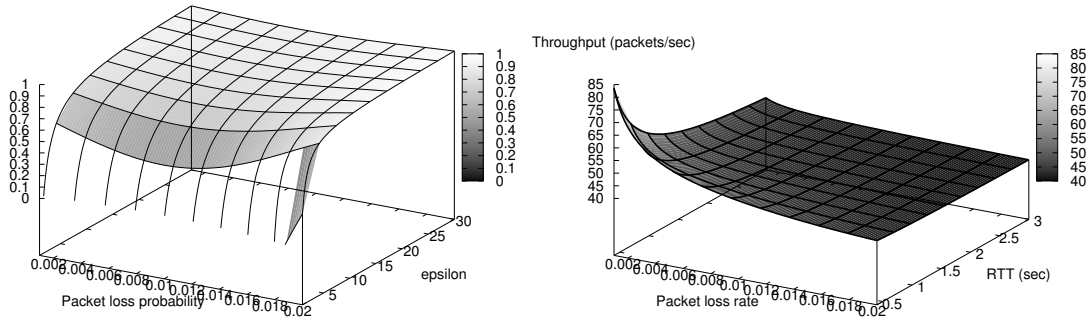


Figure 34: Simulation topology for WLAN handoff experiments.



(a) Probability to meet the latency bound ϵ

(b) Throughput prediction for TFRC as a function of RTT

Figure 35: Analytical results for TCP and TFRC during handoff with Mobile IP as a function of the packet loss rate. Parameters: $RTT^{(1)} = 400$ ms, $RTO_0^{(1)} = 800$ ms, $RTT^{(2)} = 800$ ms, $RTO_0^{(2)} = 1600$ ms, $W_0 = 1$ segment, $s = 1000$ bytes, $W_{max} = 4$ MB.

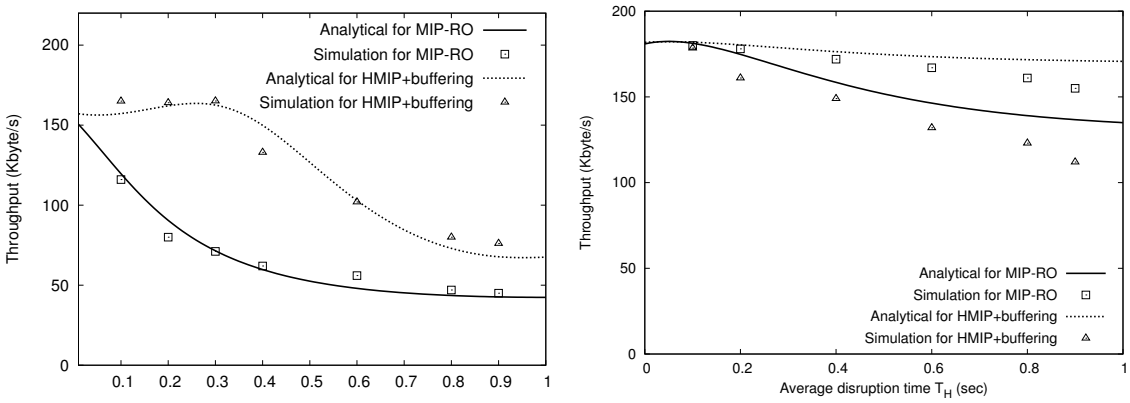
management is controlled by a relatively slower process like mobile IP, an application will not suffer dramatically. As ϵ grows, which means that the upper and lower bounds become more elastic, the probability to receive a number of packets is relatively insensitive to the packet loss rate.

Further analytical results for TFRC are presented in figure 35(b). In this figure, we show the throughput estimate of the proposed model for TFRC, when Mobile IP was the

underlying mobility protocol as a function of the handoff induced packet loss probability and the RTT on the new link. This figure clearly indicates the disadvantage of baseline mobile IP, which results into very fast throughput degradation with a slight increases of P_H.

5.3.2 Results for HMIP and MIP-RO

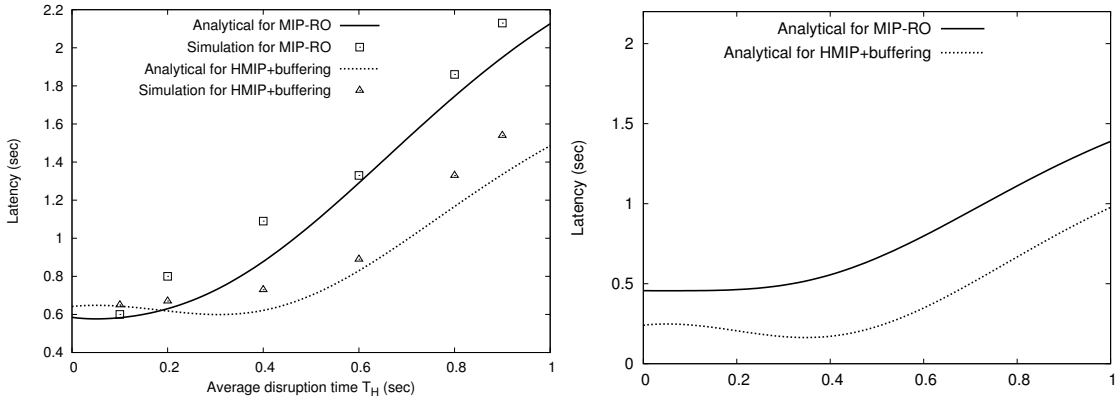
With this experiment, we want to evaluate the effect of the handoff disruption time X, on the throughput and latency of a session between the server and the client. As expected, even with MIP-RO, TCP throughput suffers considerably when disruption time is increased. The proposed models in this chapter, predict a logarithmic decrease in throughput as the packet loss rate is increased, which of course depends on the duration of the disruption. Concerning the throughput for the combination of HMIP/buffering, we can observe in figure 36(a) that TCP throughput remains pretty stable until the point where the disruption time comes close to the average RTT of the end-to-end session. Packets are buffered in the old AP and forwarded to the new AP, but after the buffer overflows, packets are dropped and the throughput decreases. This means that after the point where the forwarding buffer is full, the throughput will continue to decrease very fast since any new packet will be dropped.



(a) TCP (b) TFRC

Figure 36: Effect of disruption time on throughput for a session with a duration of 15 seconds. Parameters: $RTT = 1$ sec, $RTO_0 = 1$ sec, $plr = 0.02$, $s = 1500$ bytes, $MSS = 1460$ bytes, $W_0 = 1$ segment, $W_{max} = 4$ MB.

In figure 36(b) we present results for TFRC throughput. TFRC uses a rate control algorithm that reacts slowly to packet loss and RTT fluctuations. In addition TFRC makes use of a packet spacing algorithm that arranges the packets in time. As shown in figure 36(b), we see that the combined MIP-RO/TFRC suffers from minimal packet losses and throughput degradation. With the addition of a buffer in the previous AP, these losses are reduced further. Also it is important to note that since TFRC spaces the packet in time, the buffer at the previous AP should not overflow frequently since it does not receive packet bursts as with TCP.



(a) TCP (b) TFRC

Figure 37: Effect of disruption time on TCP and TFRC latency for a session with a duration of 10 seconds. Parameters: $RTT = 200$ ms, $RTO_0 = 400$ ms, $plr = 0.02$, $s = 1500$ bytes, $MSS = 1460$ bytes, $W_0 = 1$ segment, $W_{max} = 4$ MB

Latency results are presented in figure 37. For the combination MIP-RO and TCP, latency is increased continuously, after the disruption time is larger than the one way end-to-end delay. This results in the first packet losses and the first retransmitted packets. With HMIP, the latency incurred due to a handoff is bounded by the latency between GFA and MH. The use of buffering eliminates a series of packet losses, reduces RTO expirations and fast retransmissions leading to reduced latency as we expected from equation 72 and can be seen in figure 37(a). However, as the disruption time increases, packet losses are observed. Results for TFRC are shown in figure 37(b). We can see that the latency for the transport of a specific data chunk, is not penalized as severely as with TCP, due to the smooth variations of the TFRC rate control algorithm.

5.3.3 Recovery Period

Recovery period is represented by the time the MH has to be out of the handoff state, so that the achieved throughput in this time period is nearly the same as that prior to the handoff. Formally, what is the x for a given X, so that:

$$lim_{t_s \to x} T(X) \equiv T(0) \tag{86}$$

The answer to this question can give us useful feedback concerning the effect of the mobility ratio on the TCP throughput. Figure 5.3.3, presents results for this experiment when base-

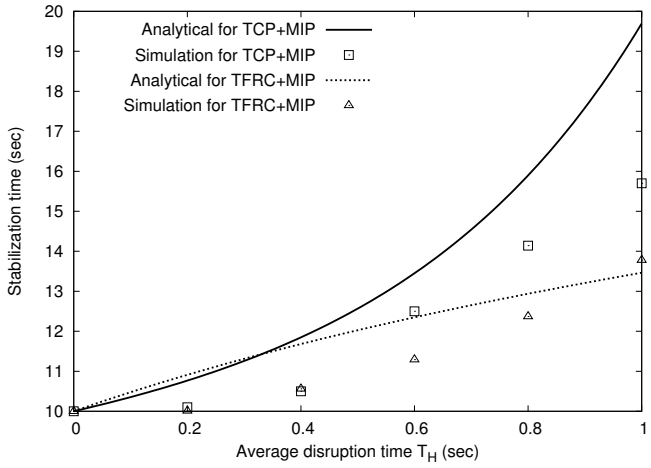


Figure 38: Required recovery time versus disruption time for both TCP and TFRC.

line Mobile IP was used for both TCP and TFRC protocols. We see that for our model the required recovery period, is increased exponentially diverging from the real measurements which are not so pessimistic. We believe that this behavior is due the interpretation of more losses as a TO indication instead of TD. However, the TFRC model does not have to classify packet loss types, allowing thus a more accurate estimation as figure 10 indicates.

5.4 Conclusions

In this chapter we presented a model for studying the effects of wireless handoffs in two transport protocols, namely TCP and TFRC. The model was found to be accurate for TCP in both the cases where HMIP and MIP-RO were used as the underlying mobility management protocols. However, the TFRC model predicts the expected throughput with

even better accuracy, due to the simpler protocol algorithms. For example the worst case error for the TCP model was nearly 22% while for the TFRC model it was 13%.

An important observation from the conducted experiments is that the use of buffering in the old access network, can significantly improved the delivered throughput. If the requirements of the system specify that no packet loss should take place, the rule of thumb for TCP, is that the buffer size should be equal to the bandwidth-delay product of the old access path times the expected disruption time. Concerning TFRC, we found that the required forwarding buffer size should be surprisingly bigger by 60% than TCP. The reason for that is the slow responsiveness of TFRC which does not drastically cut its rate, resulting in the need of a larger buffer.

We also introduced in this chapter the notion of the "recovery period", which is defined as the time required for the transport protocol to achieve the nominal throughput in a new link, after a handoff that lasted X. The slow-responsive rate control algorithm of TFRC, requires less time in order to recover when compared with TCP. However, we found that as the disruption time is increased, TFRC suffers from more packet losses than TCP, due to the slow-responsive algorithm, which is persistent on sending new packets to the network.

CHAPTER VI

VIDEO STREAMING IN HETEROGENEOUS MOBILE WIRELESS NETWORKS

Mobile wireless networks today are characterized by a high level of heterogeneity and diverse application scenarios. Most of the media streaming systems usually account for the effect on performance of either the wireless link, or the wireline Internet, or host mobility. In this chapter we move one step further, and we develop an analytically-driven video streaming protocol, suitable for heterogeneous wireless networks where both handoffs and random wireless errors are possible. Our approach is based on the development of accurate latency models for the two transport protocols that concern us: TCP and TFRC. Subsequently, our comprehensive model is used for driving the development of a protocol for managing an end-to-end video streaming session in a heterogeneous wireless environment. Main operation of the protocol is the estimation of several parameters that capture the behavior of the underlying transport protocol and the playback process at the mobile client. These parameters are then used for selecting the optimal playback buffering at the client and packet scheduling at the server.

In the second part of this chapter, we propose a new media-aware soft-handoff protocol, which when it is combined with the previously developed streaming protocol, it can assure even better QoS for a media streaming session. With the development of this complete protocol suite, we demonstrate that the use of analytical, closed-form models that capture the effect of heterogeneous wireless networks, can be utilized by a practical cross-layer optimized protocol that controls a unicast streaming session. We prove with extensive experimental and simulation results, that the algorithms of the proposed protocol, can provide better video quality at a mobile client in a heterogeneous wireless environment.

6.1 Introduction

The need for rich media services in the next generation heterogeneous mobile networks, has driven the development of several new protocols that follow different approaches in order to achieve this goal [110]. Video streaming is one of the applications that drive and will continue to drive the need for improved mobile services. Media streaming can be realized with several underlying technologies that currently exist in the wired Internet. One of the dominant media delivery systems for unicast or multicast applications today, is the client/server model. Despite however, the success of this service model in the wireline domain, in the case of wireless networks, media services have to face additional problems like the time-varying error rate and the fluctuating bandwidth. In addition, the heterogeneity of the several existing access networks today complicates even more this task. Therefore, it is important for a media delivery system to address these issues in the case of mobile networks in a systematic way that carefully considers all the necessary parameters.

The most interesting method of video delivery is through streaming, where the video server gradually transmits to the client part of the media stream [110]. Figure 39 presents a simplified view of a mobile streaming architecture in a heterogeneous wireless network. The realization of a true end-to-end architecture that optimizes the performance hot-spots in this complex inter-network, is a real challenge due to the inherent heterogeneity of the system. The literature contains several studies for media delivery platforms and video streaming especially for the Internet [2, 117, 33, 64]. However, their main feature is that they specifically focus on improving performance for a single parameter that affects performance in the end-to-end path. Therefore, in order to perform efficiently the media delivery service in terms of throughput, latency, jitter, and resource utilization, several factors that affect the end-to-end performance, have to be considered carefully, and incorporated to the streaming protocol if this is possible.

This chapter represents an effort to unify the results of the previous chapters, and proceed with the formalization of media delivery in heterogeneous wireless networks, through consideration of the several parameters that affect this process. We develop performance models that provide a balance between accuracy and simplicity. Simplicity is necessary for

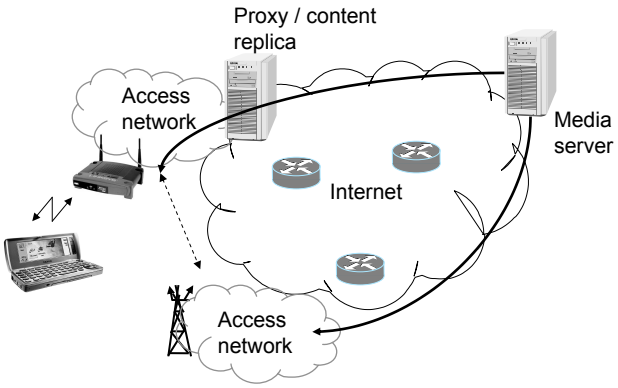


Figure 39: Simplified wireless media streaming architecture.

the use of the analytical closed-form formulas for online use by a media streaming protocol. Our goal is that after the formalization of the problem, we will be able to answer these questions: What is the most efficient transport protocol to use for a wireless mobile network which exhibits certain characteristics? Given that we can distinguish between packet loss that happens due to handoffs or due to wireless errors, how can a streaming protocol can be engineered? What are the options that an application has given that it uses a specific transport protocol like TCP, TFRC or the new SCTP?

6.2 Related Work

The area of error-resilient wireless video transmission has attracted significant research in the last years due to the high demand for media application in wireless devices. Mechanisms like forward error correction (FEC) [39, 58], ARQ [119], and video unit interleaving [109], have been extensively researched and reside at the forefront of the tools for error-resilient wireless video transmission. However, the additional requirements of reduced latency and the sensitivity of hybrid coded video to errors, has pushed the research into applying even

more strong interaction between the network protocol stack and the video application [110]. Therefore, several cross-layer techniques may be employed in order to communicate channel status to the application, so that the appropriate actions can be taken [2, 29, 27]. Several wireless content delivery approaches have focused on MAC layer optimizations of the wireless link that resides at the last hop of an end-to-end connection [29]. Another important class of mechanisms for error-resilient video transmission, is based on middleware-based systems, where a mobile proxy is strategically located at the boundary between the wired and wireless networks. The proxy server, which is usually located at the base station, handles ARQ requests and tracks errors. In the case where no error occurs in the wireless link, the video ARQ proxy server acts like a router that routes incoming packets to the wireless link. If an error occurs in the wireless link, the proxy will resend locally the packet as soon as the bandwidth budget allows it. Several of these proxy-based mechanisms can be found in [13, 10, 110].

On the transport protocol side, a few techniques have been developed mainly for TCP, and recently for TFRC, in order to improve performance over wireless channels [10, 24, 3, 30, 43]. These methods either hide packet loss caused by wireless channel errors, or provide an end host the ability to distinguish between packet loss caused by congestion and that caused by wireless channel errors. For example in [24], the authors develop a method for differentiating between congestion and wireless induced losses. Another interesting approach is the UDP-Lite protocol, which is tailored to wireless environments where bit errors take place [63]. The use of rate control algorithms for wireless channels was shown to be able to deliver increased throughput with protocols like WTCP [96]. WTCP is rate-based, and uses only end-to-end mechanisms, performs rate control at the receiver, and uses inter-packet delays as the primary metric for rate control. Recently the analytical rate control (ARC) scheme, was proposed for wireless applications that require smooth rate fluctuations [3]. The important feature of this protocol, is the derivation of analytical model which describes the TCP throughput. Subsequently, the ability of the wireless receiver to distinguish wireless packet losses, is used to drive the correct behavior of the rate control algorithm, which reacts only to congestion events. Another interesting approach can be seen in [30], where

the authors propose a rate control scheme called MULTFRC. The basic idea behind this protocol, is to measure the round trip time, and adjust the number of end-to-end TFRC connections so that the wireless bandwidth is utilized efficiently.

Apart from the developed techniques that combat wireless errors, an additional challenge that has to be encountered, is that of handoffs due to mobility. The main mechanism that was developed in order to resolve the problem of mobility management for IP networks is Mobile IP [83]. The usual way of handling handoffs is through buffering either at the base stations [108, 29], or at the mobile client [55]. One system that we are aware of, and performs media aware handoff in WLANs, is presented at [20, 19]. However, the proposed system is based heavily on modifications of the media servers, the access network, and the wireless media clients. It is understandable that an approach like this incurs significant implementation overhead. On the other hand, the effect of cellular handoffs in media performance is studied in [66], where the authors propose new bandwidth allocation schemes to alleviate wireless bandwidth fluctuations. We are only aware of one recent work that evaluated the performance of a streaming application for the simple case of WLAN to WLAN handoff [56].

6.3 System Model

This section presents the various aspects of our system model that we used throughout this chapter. First goal towards a comprehensive performance model, is the definition of system model that takes into account several parameters that can affect video streaming performance. We adopt a path model that is able to capture the variety of scenarios and uses cases that we want to model. Figure 40 presents a high level view of our system model. The application scenario assumes that there is a unicast end-to-end session with the data flowing from the correspondent host (CH) to the mobile host (MH). According to a typical mobility scenario, the MH is initially associated with the first access network (AN_1), and at some time in the future it may move towards AN_2. The two access networks are modeled separately, while they are both attached to the core network (CN) which in general can be the Internet. One initial assumption that we make, is that while the mobile host is

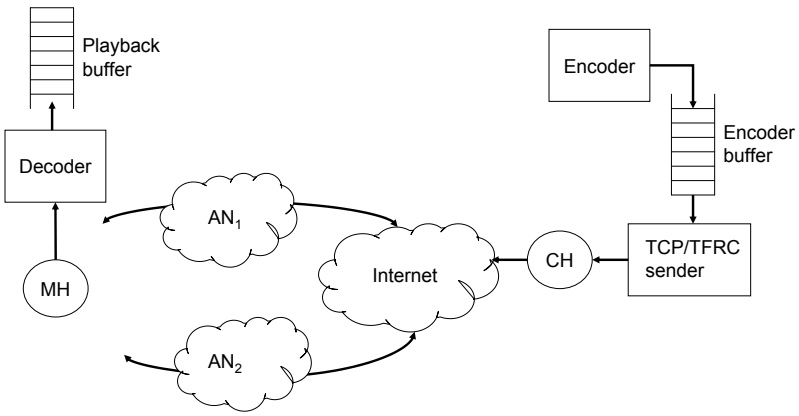


Figure 40: System model for joint transport protocol and video streaming characterization during handoffs.

connected to AN_1, the transport protocol session is in steady state. Subsequently, after handoff is performed to AN_2, the transport protocol will reach the steady state at some time in the future.

Concerning the core network, it is modeled as a two-state Markov chain, an approach that has been shown to predict quite well the Internet packet loss behavior [82]. The access networks are modeled as three-state Markov chains, where the three states are good, bad and handoff ($G = 0, B = 1, H = 2$). This implies that the handoff induced packet losses are distinguished from other packet losses due to wireless errors in the access network. The duration of the successive good states, $\{G_i, i = 1, 2, ...\}$ are independent and identically distributed (i.i.d) with an exponential cumulative distribution function with mean $E[G_i] = T_G$. The durations of time in the bad wireless state, is denoted by $\{B_i, i = 1, 2, ...\}$ are i.i.d with mean $E[B_i] = T_B$, and independent of the $\{G_i\}$. The same holds for the handoff state, $\{H_i, i = 1, 2, ...\}$, which is again assumed to be exponentially distributed with mean $E[H_i] = T_H$. Therefore, the transition probability matrix for the access network will be:

$$M_{AN} = \begin{pmatrix} \pi_{GG} & \pi_{GB} & \pi_{GH} \\ \pi_{BG} & \pi_{BB} & \pi_{BH} \\ \pi_{HG} & \pi_{HB} & \pi_{HH} \end{pmatrix} \quad (87)$$

Note that the adoption of a different path model will not alter the analytical formulas developed in this chapter, but it may affect the accuracy of the derived packet loss rate. Even so, this path model takes into account the two possible cases that may lead the AN to be in bad state, and these are due to handoff or wireless errors. From this discussion, it is evident the packet losses are decomposed into many elements according to the reason of the packet loss. In reality it will be probably difficult to measure and distinguish end-to-end packet losses according to their origin. Therefore, our objective is to use estimates of the packet losses in order to derive an approximation of the end-to-end latency.

6.4 Performance Analysis Model

In this section we analyze the decomposed end-to-end path model and estimate all the crucial parameters that will affect the delivery of a media stream for a specific transport protocol. Objective of our performance model is the evaluation of the latency for the TCP or TFRC transport protocols. Since central part in any media communications system is the use of a playback buffer, we subsequently consider the behavior of the playback buffer at the mobile client as a function of the end-to-end latency estimate. Finally, we include in the model the precise effect of mobility protocols on the ultimate performance of the media delivery system. With this modular approach we successively build a model that accommodates more and more parameters of the heterogeneous wireless network shown in figure 39.

6.4.1 Latency

An important concern when developing stochastic models is the validation of the stationarity assumption [89]. In our case, the stationarity assumption simplifies the calculation of the random variable that describes the latency, since both the mean and variance of the latency will have a constant value. We make this assumption for the end-to-end latency so that an

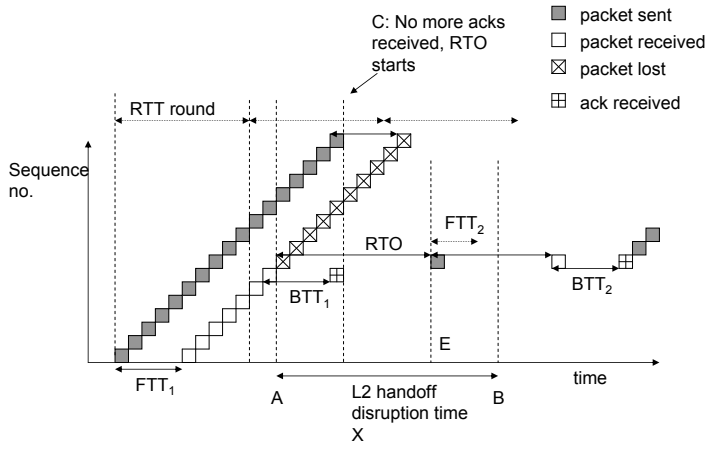


Figure 41: Packet-level TCP behavior at the sender during IP layer handoff.

analytically tractable model is obtained. So initially, before we calculate the average value for the latency, we have to estimate the packet loss rate in the end-to-end path.

Wireless packet loss rate (P_W): We start by the estimation of the packet loss rate due to wireless errors in the access networks. We adopt the Gilbert path model for capturing wireless channel behavior, since it is simple and fairly accurate [8]. Therefore, the average packet loss rate due to wireless errors will be given by:

$$P_W = \frac{\pi_{GB} + \pi_{HB}}{\pi_{GB} + \pi_{BG} + \pi_{HB} + \pi_{BH}} \tag{88}$$

The transition probabilities are calculated using maximum likelihood estimators [18]: $\hat{\pi}_{GB} = \frac{n_{GB}}{n_G}$, where n_{GB} is the number of times in the ACK messages that B state follows G state, n_{BG} is the number of times G follows B, and n_G is the number of times a good state is followed by a good. Concerning the calculation of n_{HB}, it can be similarly defined.

Handoff packet loss rate (P_H): Figure 58 will be used for explaining the behavior of TCP during a handoff event. The variables FTT_1/BTT_1, and FTT_2/BTT_2 describe the

forward and backward trip times for the two access networks respectively. According to this figure, at time instant t_A, layer 2 connection is lost and at time t_A+RTO, the TCP sender retransmits the first lost packet. Now if $t_B > t_C + FTT_1$, then clearly no duplicate acknowledgments will be received at the sender, and the only way for TCP to resume the data flow is by expiration of the RTO of the first lost packet. On the other hand, as the disruption time X shrinks, and if $t_B \leq t_C + FTT_1$ then the probability to receive a number of the last packets (close to time instant t_C) is increased. If this happens, then the sender would fast retransmit the first lost packet, resulting into a faster recovery. If we rewrite the previous equation we have (and because $t_C = t_A + BTT_1$): $t_B \leq t_C + FTT_1 \Rightarrow X \leq RTT_1$. Therefore, the probability of handoff induced packet loss is the probability of that the one way latency L, is smaller than the handoff duration X, i.e. $P_H = P[L < X]$. If we decompose the latency to the two components from which its consists, then the previous equation is written:

$$P_H = P[L_N + L_{protocol} < X] \tag{89}$$

The two variables L_N and $L_{protocol}$ express the latency induced by the network and the transport protocol respectively. We define as f_{L_N} the distribution of the end-to-end network induced latency. On the other hand, the disruption time X is a parameter that depends of the mobility management a protocol. In general it can have a fixed value for a specific protocol [114]. In this chapter we assume that is is exponentially distributed with mean T_H. Since there is a need to estimate the end-to-end latency, we will proceed to find it next.

Steady state latency distributions: The distribution of the TCP latency can be in the general case heavy tailed [22], since the use of the best-effort packet forwarding service, that the core of the Internet supports cannot provide any delivery guarantees. However, its precise p.d.f. will depend on the assumptions we make about the packet loss model. Let now R and Y denote the random variables of the RTT and the RTO respectively. From figure 58 we can see that when $L < X$ two more cases arise:

Case 1, $X > t_E - t_A$: This condition means that the MH will still be in the handoff

state, while the RTO expires the first time (time instant t_E in figure 58). So if we want to find the probability that the AN will be in handoff state for time ϵ after handoff, then this can be expressed as:

$$P\{S(t_E+\epsilon)=G|S(t_E)=H\}=P_g(\epsilon)$$

The above holds due to the memoryless property of the distribution of X, causing thus the channel state at time plus ϵ, to be independent from the state at t_A. Therefore:

$$P_g(\epsilon)=P\{S(\epsilon)=G\}=\frac{T_G}{T_H+T_G+T_H}(1-e^{-\epsilon/T_G}) \tag{90}$$

The term $P_g(Y)$ expresses the probability for the channel to be good state after time Y, when the RTO expires. So the average TO duration, for every possible RTO value, will be equal to the probability that the channel is in good state at that specific RTO (given that it was bad before) times the value of the RTO. So this value will be:

$$L_{TO}(Y)=\sum_{i=1}^{6}2^{i-1}Y\times P_g(iY)\prod_{j=0}^{i-1}(1-P_g(jY))+64YP_g(64Y)\prod_{i=0}^{6}(1-P_g(iY)) \tag{91}$$

The product term in the previous equation expresses the probability that the AN was in handoff state, when the TO expired in the previous time instants before i. In addition, after the first six consecutive TOs the value of the TO will be fixed to $64Y$ [99]. The last term on the above equation captures this effect.

Case 2, $X<t_E-t_A$: In this case, we can see from figure 58, that it will also be $t_C<t_E$. This means that the sender will not experience a TO, but instead a TD and so it will fast retransmit the first missing packet. The average latency introduced due to this event be:

$$L_{TD}=\int_{t=0}^{T_H}tP_g(t)dt \tag{92}$$

Therefore, the total latency for TCP can be expressed from the previous two equations:

$$L_H^{TCP}=\sum_{l=0}^{\infty}P[l<L]=\int_{t=Y}^{T_H}L_{TO}(t)f_X(t)dt+\int_{t=0}^{T_H}tP_g(t)dt \tag{93}$$

The first term in the above equation follows from the assumption that the disruption time due to handoff X, is exponentially distributed with a mean equal to T_H and a p.d.f $f_X(t)$.

Concerning the core network induced delay L_N, that occurs mainly due to buffering at the routing infrastructure, it has been shown that it could be modeled as a shifted Gamma distribution [25, 64]:

$$f_{L_N}(t) = \begin{cases} \frac{\lambda e^{-\lambda t}(\lambda t)^{\nu-1}}{(\nu-1)!} & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \tag{94}$$

We will follow this latency distribution in this chapter also so that the derivation of an analytical closed form solution is possible. Several possible analyses can be preformed in order to model more accurate the core network performance, but this research is out of the scope of this dissertation.

6.4.2 Joint Latency and Playback Buffer Model

The expected delay of packets due to handoff is expressed as L_H, while the probability for a packet to miss its deadline is P_H. These two quantities are primarily related to the handoff disruption time X. The important point that we should stress here, is that the packets that were delayed due to handoff and missed the playback deadline, are the packets that were temporarily buffered at the old access network. In that case $P_H \simeq 0$, since the purpose of the forwarding buffer is to eliminate handoff packet losses. Nevertheless, buffering cannot help if the disruption time X is long, resulting into the delivery of useless packets to the media application. Therefore, the probability for a packet to be delayed, excluding the case of handoff, can be formulated as:

$$\begin{aligned} P_D &= P[\text{sent time} + \text{latency} + \frac{\text{buffer occupancy}}{\text{playback rate}} \leq \text{deadline}] \\ P_D &= P[t_s + L_N + L_{protocol} + \frac{b_1}{c_1} \leq t_d] \end{aligned} \tag{95}$$

where t_s is the time that a video unit was sent, while t_d is the playback deadline for this unit. For TCP and TFRC this expression will be different. TFRC does not introduce any latency since it does not control retransmissions and so $L_{TFRC} = 0$. In the case of TCP, L_{TCP} will be the latency incurred by the TCP retransmission mechanisms, due to the packet loss rate P_N in the core network. Therefore equation 115 will be:

$$
\begin{aligned}
P_D^{TCP} &= P[t_s + L_N + P_N L^{TCP} + \frac{b_1}{c_1} \leq t_d] \\
P_D^{TFRC} &= P[t_s + L_N + \frac{b_1}{c_1} \leq t_d]
\end{aligned} \tag{96}
$$

Now the probability for a packet to be delayed plus the case of handoff, and when the playback buffer is considered, can be written based on the previous equation:

$$
\begin{aligned}
P_{DH}^{TCP} &= (1 - P_D^{TCP}) P[t_s + L_N + P_H L_H^{TCP} + \frac{b_1}{c_1} \leq t_d] \\
P_{DH}^{TFRC} &= (1 - P_D^{TFRC}) P[t_s + L_N + P_H L_H^{TFRC} + \frac{b_1}{c_1} \leq t_d]
\end{aligned} \tag{97}
$$

These values will later help us quantify the model performance in terms of correct deadline estimation at the sender. By elaborating on the previous formulas we can write for TCP:

$$
P_D^{TCP} = P[L_N + P_N L^{TCP} \leq a] = \int_{-\infty}^{+\infty} F_{L_N}(a - y) f_{L^{TCP}}(y) dy \tag{98}
$$

with $a = t_d - t_s - \frac{b_1}{c_1}$. And so for P_{DH} we finally have:

$$
\begin{aligned}
P_{DH}^{TCP} &= (1 - P_D^{TCP}) P[L_N + P_H L_H^{TCP} \leq a] \\
&= P_D^{TCP}[L_N + P_H L_H^{TCP} \leq a] \int_{-\infty}^{+\infty} F_{L_N}(a - L_H - y) f_{L_H^{TCP}}(y) dy
\end{aligned} \tag{99}
$$

The distributions in these formulas are simple sums of exponential values, making thus the derivation of a final solution straightforward.

6.4.3 Effect of Mobility Protocols

A novel aspect of our streaming system, is the consideration of handoff between asymmetric wireless access networks, as we showed earlier in figure 39.

Forwarding buffer packet loss rate (P_{BU}) and latency (L_{BU}): As a first step we would like to account for the case where the mobility management protocol employs a forwarding buffer at the old AN for salvaging in-flight packets. The effect that the buffering mechanism will have, is that several packets will not be lost, but a prolonged handoff may

render them useless since they might miss the playback deadline. Note that another type of non real-time traffic would not suffer from this side-effect.

If we assume that the server where the buffering is performed, is modeled as an M/M/1/K queue [16], with finite capacity K and average load ρ, then the packet loss probability due to handoff has to be recalculated. The packet loss probability in this case is equal to the conditional probability that a packet arrives faster in the access network, given that the buffer is full:

$$P_{BU} = (1 - P_{DH})P[L_N + L_{protocol} < X | S = K] \tag{100}$$

By elaborating on the previous equation it will give:

$$\begin{aligned} P_{BU} &= (1 - P_{DH})P[L_N + L_{protocol} < X] f_S(K) \\ &= (1 - P_{DH}) f_S(K) \int_{-\infty}^{+\infty} F_{L_{pr}}(y) f_X(y) dy \\ &= (1 - P_{DH}) f_S(K) \int_{-\infty}^{+\infty} f_X(y) [\int_{-\infty}^{+\infty} F_{L_{protocol}}(y - z) f_{L_N}(z) dz] dy \end{aligned} \tag{101}$$

In the previous equation, f_S is the c.d.f. of the random variable S that describes the forwarding buffer occupancy. So equation 101, describes in a parametric form, the packet loss probability directly attributed to handoff. Concerning the average latency due to the forwarding buffer, it will be equal to the average time a packet will spend in the forwarding buffer [89]:

$$L_{BU} = \frac{1}{P[S = K]} \frac{\rho(1 + K\rho^{K+1} - (K + 1)\rho^K)}{(1 - \rho)(1 - \rho^{K+1})} \tag{102}$$

and by using equations 93 and 102 the total latency is equal to:

$$L_{TBU} = L_{BU} + L_H \tag{103}$$

In the previous equation, one more value that can be added is the one that captures the delay that a specific mobility management protocol introduces. We calculate this value for mobile IP, hierarchical mobile IP, and mobile IP with route optimization in appendix B.

Optimal playback rate and buffer size: Based on the derived equations, we can define the optimal playback rate c_1 given the current playback buffer occupancy b_1. In order for the playback buffer not to underflow the following condition must then hold for TCP:

$$\frac{b_1}{c_1} > \lfloor \frac{min(T_H, L_{DH}/L_{TBU})}{RTT} \rfloor RTT \tag{104}$$

6.5 Protocol for Unicast Media Streaming in Heterogeneous Networks

After developing an analytical model that characterizes the performance of TCP and TFRC in heterogeneous wireless scenarios, we have to analyze how the developed model can be utilized in order to drive the behavior of a media streaming protocol. In general, the following functions can be part of a streaming protocol: rate estimation, error control, feedback, packet scheduling, playback. For TCP, the first three functions are part of the protocol specification, while the last two may change and be implemented by the application. For TFRC only the rate estimation algorithm is part of the protocol specification. Now, the proposed heterogeneous wireless network streaming protocol (HWNSP) control packet scheduling and playback. The protocol defines a set of functions that roughly distinguish the operation into two phases, namely initialization and steady phase. Initial objective of the protocol is to identify the state of the end-to-end path, which can be good, bad, or handoff. Subsequently the protocol proceeds with the estimation of several parameters that are related with the state of the channel. The previously developed analytical model helps with the estimation of the necessary parameters, so that the aforementioned decisions can be made efficiently. The detailed protocol operation both at the client and the server is discussed next.

Client operation: The algorithm that the client executes throughout the streaming session, can be seen in figure 42. During initialization, the client requests from the server a media file, and also informs the server about the desired playback rate c_1, and initial pre-roll delay Δ. This operation can be performed with either the SIP or the RTSP protocols [93]. During the steady phase, the algorithm initially updates the estimate of the forwarding

initialize()

1: initialize c_1, Δ
2: $send\_msg(server, clip\_name, \Delta, c_1)$

steady_phase()

1: $K \leftarrow cwnd$, $c_1 \leftarrow$ // equation 104
2: $\hat{L}_{CN} \leftarrow$ // equation 114
3: **if** HANDOFF **then**
4: $\quad \hat{T}_h \leftarrow (handoff[types] \rightarrow delay)$
5: $\quad \hat{P}_H \leftarrow$ // equation 109
6: $\quad \hat{L}_{TBU} \leftarrow$ // equation 103
7: $\quad \hat{P}_W \leftarrow$ // equation 108
8: $\quad c_1 \leftarrow$ // equation 104
9: $\quad send\_msg(server, \hat{T}_H, \hat{P}_H, \hat{P}_W)$
10: **end if**
11: **if** BAD WIRELESS **then**
12: $\quad \hat{\pi}_{GB} = \frac{n_{GB}}{n_G}$...
13: $\quad \hat{P}_W \leftarrow$ // equation 108
14: $\quad c_1 \leftarrow$ // drop playback rate, equation 104
15: $\quad send\_msg(server, \hat{P}_W, c_1)$ // according to the TCP or TFRC allowed rate
16: **end if**

Figure 42: Protocol operation at the client.

buffer capacity (line 1), and in addition configures the playback rate if the network latency $\hat{L}_N$ changes. The client can be informed for a handoff decision from the lower protocol layers (L2/L3). When this happens, the client enters handoff state, and immediately calculates the necessary model parameters (lines 5-8) namely $\hat{T}_H$, $\hat{P}_H$, and $\hat{L}_{TBU}$. By estimating this values, the client knows the effect that this handoff will have on the ability of the TCP protocol to deliver packets. Therefore it sets the playback rate appropriately as line 10 indicates, according to equation 104. When the connection with the new link has been established, the client can inform the server about the reason of the disruption (line 11).

Server operation: The server operation can be seen in figure 43. The protocol operation at the server is also distinguished into the same two phases. During initialization, the server sets the estimated forwarding buffer size K of the AN to be equal to the bandwidth delay product (congestion window) of the first path. It also sets the value of the estimated disruption due to handoff (X) equal to zero, and the average load of the server that performs forwarding to 0.8. During steady phase, the server continuously updates several

initialize()

1: $\rho \leftarrow 0.8$, $\tilde{T}_H \leftarrow 0$
2: $K \leftarrow cwnd$

steady_phase()

1: $K \leftarrow cwnd$ // update estimates
2: calculate Gamma distribution parameters
3: $\tilde{L}_{CN} \leftarrow$ // equation 114
4: $rtp\_packetize(VDU)$
5: $send(VDU)$
6: $detect\_handoff()$
7: **if** HANDOFF **then**
8: // Server detected handoff
9: $\tilde{P}_{HB} \leftarrow f(C, P_H)$ // equation 101
10: $\tilde{L}_{HB} \leftarrow f(C, P_H) + \tilde{T}_H$ // equation 102
11: // TCP should not reduce *cwnd*
12: $\tilde{c}_1 \leftarrow$ // equation 104
13: $\tilde{D} \leftarrow$ // calculate distortion if real-time streaming
14: $schedule(send\_buffer, \tilde{D}, \tilde{L}_{HB}, \tilde{c}_1)$
15: **end if**
16: **if** BAD WIRELESS **then**
17: $\tilde{\pi}_{GB} = \frac{n_{GB}}{n_G}$... // estimate transition probabilities
18: $\hat{P}_W \leftarrow$ // equation 108
19: $\tilde{c}_1 \leftarrow$ // drop playback rate equation 104
20: $send\_msg(server, \tilde{P}_W, \tilde{c}_1)$ // according to the TCP or TFRC allowed rate
21: **end if**

$detect\_handoff()$

1: $correlation\_test(RTT_{current}, RTO_{current}, \tilde{T}_H, \tilde{L}_N, \tilde{L}_{HB})$

Figure 43: Protocol operation at the server.

parameter estimates while streaming to the mobile client. These are the estimates for K, and the expected network latency $\tilde{L}_N$. The Gamma distribution parameters are defined in equation 114. The interesting part of the server operation arises in case of a handoff. In this case the server is also estimating $\tilde{L}_{HB}$, and subsequently estimates the client playback rate c_1 as $\tilde{c}_1$. This is an important function of the protocol, since the server is able to approximate the rate at which data are removed from the playback buffer at the mobile client. As a next step, the server schedules new packets at a lower rate in order to minimize the expected number of dropped packets because of the handoff.

Client-oriented protocol: Even though the server-oriented version of the protocol might assure a near-optimal packet scheduling, it might not be always possible to upgrade

Table 3: Vertical handoff latencies between heterogeneous wireless networks.

Handoff type	Latency (sec)
WLAN⇒GPRS [42]	1.67
GPRS⇒WLAN [42]	1.281
WLAN⇒CDMA2000 [81]	1.01
CDMA2000⇒WLAN [81]	0.6
WLAN⇒UMTS [107]	1.3
UMTS⇒WLAN [107]	0.7

the server with the proposed protocol since it might not scale as the number of clients is increased. Therefore, a version of the protocol that operates only at the client will consist of the set of operations defined in the algorithm in figure 42. The only difference is that the client need not to inform the server for the local parameter estimates. Experimental results that depict the relative merits of each approach will be given in a later section.

Handoff latencies: Table 3 provides a summary of the values that can be used by the proposed protocol, when handoff between a heterogeneous networks takes place. Clearly these values depend on the infrastructure used, network configuration etc [42, 81]. However, they provide good typical estimates concerning a specific network's handoff latency. The total handoff latency is the sum of all individual latencies which include the latencies for link-layer handoff, movement detection, and registration.

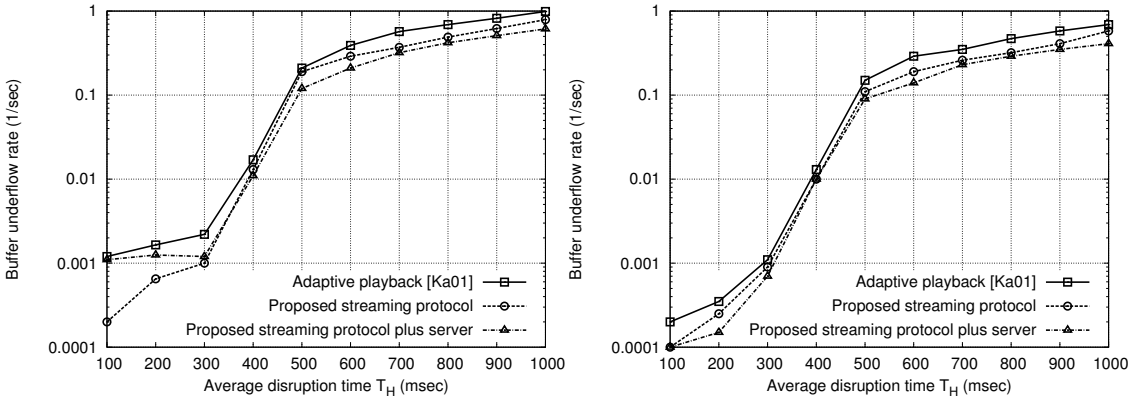
6.5.1 Experiments

The network testbed for our experiments consists of a client/server configuration that are linux boxes while the middlebox is freeBSD machine that acts as a router. We used the middlebox with the Dummynet software [36], for emulating the packet losses due to wireless errors, handoffs between access networks, and buffer overflows in the core network routers. The sequences FOREMAN, AKIYO, and COASTGURAD [94], were used for the video streaming experiments. The H.263 encoder [44] was used for encoding the YUV sequences into various bitrates. The video units were packetized into RTP packets and the sent to the transport protocol which is our case were TCP and UDP/TFRC. Due to the short duration of the sequences (150 frames), they were repeatedly fed as input to the encoder, so that

encoded sequences of longer duration could be obtained. The capacity of the bottleneck link between the two routers was set to 250Kbps. All the sequences were encoded at 128Kbps with a target frame rate of 15 fps. The results were obtained by running the same scenario 100 times and averaging the PSNR values of the same experiments. A summary of all the parameters used throughout our experiments in this section, can be seen in table 4.

Table 4: Model parameters used for the video streaming experiments.

Network Parameters		Protocol parameters	
$AN : T_H$	(WLAN) 50 ms	T_0	200 ms
$AN : T_G$	(WLAN) 1000 ms	W_{max}	6 MByte
$AN : T_B$	(WLAN) 100 ms	W_0	1 segment
$CN : T_G$	1000 ms	MSS	1460 bytes
$CN : T_B$	5 ms	Mobile speed	1-20 m/s

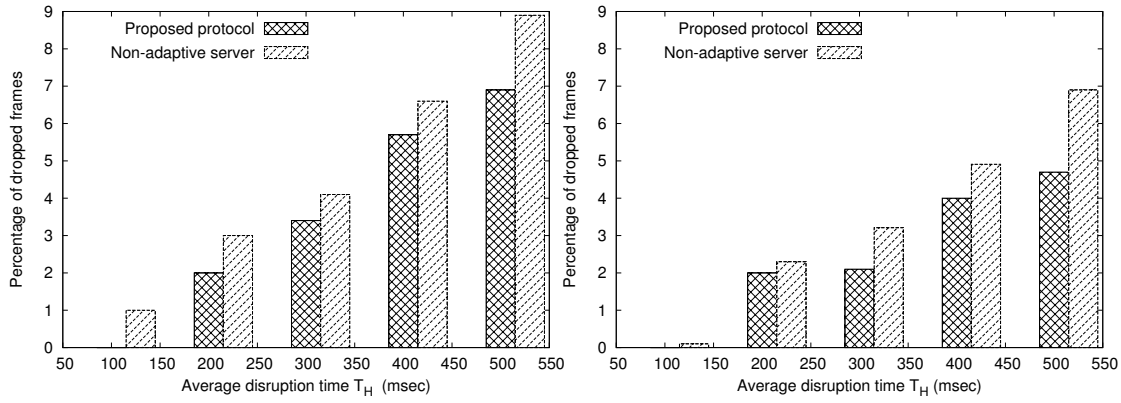


(a) TCP (b) TFRC

Figure 44: Buffer underflow rate for WLAN→WLAN handoff.

In the first experiment, we evaluated the effect of the handoffs on performance of the playback buffer. In figure 44, the x-axis depicts the average value for the handoff duration (T_H), and in the y-axis the normalized buffer underflow rate. We compare our approach, with a playback adaptation strategy reported at [54], and we also set the playback buffer size to 200 packets. Results for TCP can be seen in figure 44(a), and demonstrate that the proposed playback adaptation strategy can lead to a lower number of buffer underflow events when compared to currently employed wireless playback adaptation algorithms [55].

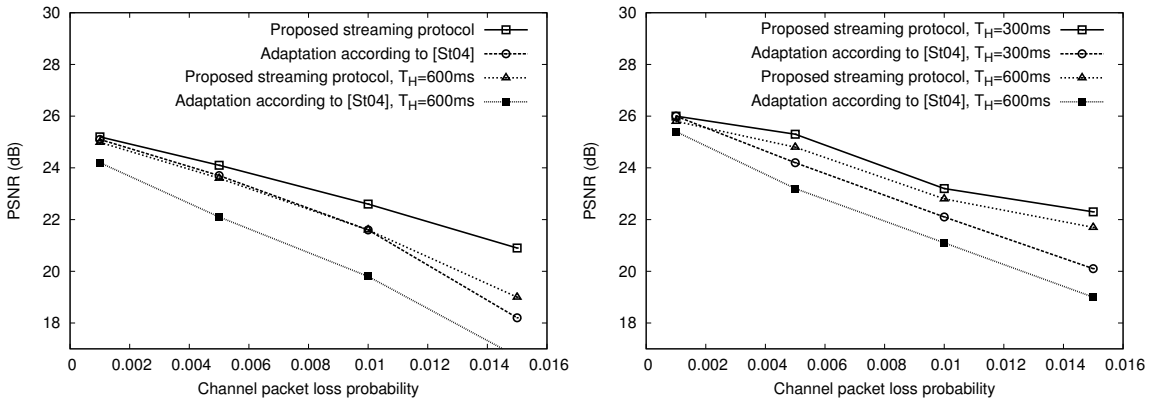
It is interesting to note that when the handoff has small duration, the version of the protocol where the server is also active, does not correspond to large benefits. However, when the handoff duration is increased the mobile client cannot make good estimates concerning the optimal playback rate. Results for TFRC are shown in figure 44(b). The proposed buffer adaptation for TFRC is proven more essential than initially expected, due to TFRC's reluctance to decrease fast the output data rate. This behavior results into a higher number of buffer underflow events for the TFRC protocol. However, when HWNSP is added to the system, the protocol commits to TFRC smaller number of packets without following TFRC's indicated allowed rate, leading thus to a reduced number of buffer underflow events.



(a) TCP (b) TFRC

Figure 45: Video quality expressed though the percentage of the dropped frames at the server.

The next set of experiments evaluates the actual video quality at the mobile client. The two metrics that we considered for this set of the experiments, is the number of dropped frames at the server, and of course the actual PSNR of the video sequence due to the handoffs experienced at the mobile client. Figures 45(a,b) depict the number of dropped frames at the server during a handoff event for TCP and TFRC respectively. The operation of the proposed protocol at the server essentially adapts to the changing network conditions by estimating the expected disruption time $\tilde{T}_H$ at the mobile client. By doing so, the server reduces the output rate, and so several frames are not sent from the server for the estimated duration of the disruption, resulting thus into the frame drop events at the server. However,



(a) PSNR for TCP transport, as a function of the core network packet loss (P_C) and P_W=0.001

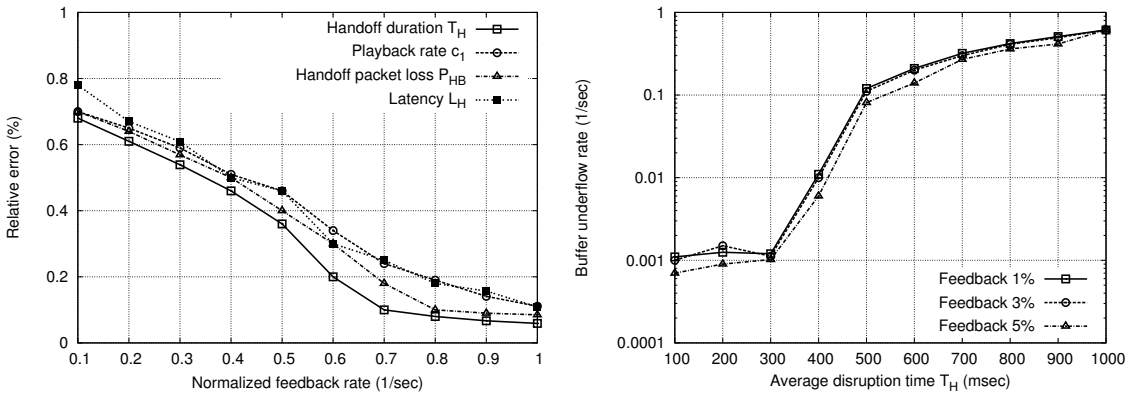
(b) PSNR for TFRC transport, as a function of the wireless packet (P_W) and P_C=0.001

Figure 46: Video quality expressed though the PSNR at the mobile client.

a typical solution like the one reported at [56], where no estimates of the precise disruption time are made, result into even more lost frames. The same situation can be observed when UDP/TFRC is used for transport in figure 45(b). The difference is that baseline TFRC will experience more dropped frames, while the addition of the proposed protocol reduces the output rate, leading to a lower number of dropped frames.

At the core of our performance evaluation, are the experiments that stress test the heterogeneous path model, and essentially, the effect of heterogonous packet losses. Results for peak signal-to-noise ratio (PSNR) are depicted in figure 46. More specifically, in figure 46(a) we show the effects for a fixed wireless packet loss rate of $P_W = 0.001$, and varying packet loss on the core network P_N. We compare out protocol with the wireless video adaptation strategy developed at [102], where the authors of that study consider only wireless errors. The expected advantage of the proposed protocol, is that adaptation is performed at a finer granularity since the behavior of the protocol should be different when handoff takes place compared to the case of random wireless errors.

Figure 47, presents results for an experiment that tests an important part of the proposed protocol functionality. This is the rate of client feedback to the server, and its effect on the accuracy on the parameter estimation process. Generally, the guideline for streaming



(a) Error estimate $\hat{e}$, for several protocol parameters as a function of feedback rate

(b) Buffer underflow rate as a function of the feedback rate

Figure 47: Effect of the client feedback rate.

sessions indicates that the rate of feedback should not exceed 5% of the total bandwidth consumed in a streaming session [110]. We followed this rule for our experiments. Figure 47(a) depicts this effect of the feedback rate on the on the parameter estimation process at the server. We can see that the estimates diverge significantly from the actual values, when the feedback rate is low. It is interesting to note in this figure that the effect of feedback is different to different parameters. For example the handoff disruption estimate $\tilde{T}_H$, and the end-to-end latency $\tilde{L}_H$, are not affected since their value does not vary significantly. After a specific threshold of feedback rate is reached, the server is able to make very good estimates for these two parameters. However, the value of the optimal client playback rate $\tilde{c}_1$, is more difficult to be predicted correctly.

Now in figure 47(b), we demonstrate the effect of the feedback rate to the actual system performance in terms of the playback underflow rate. The increase in the feedback rate contributes to the better estimation of the needed parameters, and consequently the observed number of buffer underflow events is lower.

6.6 Proactive Soft-Handoff of Media Flows

In this section we will show that the use of an end-to-end handoff management protocol, can improve bandwidth utililization and reduce jitter for media flows, when handoff takes place. The proposed handoff management protocol is general enough in the sense that it

operates over generalized connectionless IP-based networks (Internet). It is implemented as part of SCTP, which supports multihoming and allows thus the existence of transport layer sessions that utilize multiple IP addresses. The set of functions that performed by the protocol during handoffs, are implemented at the end points, so that a completely decentralizes solutions is obtained.

6.6.1 Handoff Algorithm

When a mobile host enters a new foreign network or subnet, it must obtain a valid IP address from the visited network (e.g. via DHCP [35] or IPv6 auto-configuration) which is called Care-of-Address (CoA) [1]. This operation is performed regardless of underlying mobility management protocol. After a valid IP address is obtained, the algorithm in figure 48 is executed. Note that when the proposed protocol is implemented on top of STCP without the use of Mobile IP, the SCTP sender sends an address configuration (ASCONF) message to the server requesting from it to add the new address to the association. If TCP was used the connection would break requiring thus the establishment of a new connection. Now after the CH adds the new address to the session, it starts using it immediately, assuming thus that the mobile host is reachable through this new address. In the meantime the media streaming protocol that we presented in the previous section (HWNSP), can star the estimation process after the soft handoff has finished. The advantage for the streaming session is that due to the soft-handoff nature of the protocol, disruption time is minimized, and a the adaptation of the streaming session to the new link is faster.

The binding update procedure is separated in two parts – A binding update that corresponds to the updates sent to the CH during handoffs, and a periodic binding update that is sent directly to the HA (line 9). The second part corresponds to a periodic update that has limited relationship with the updates due to host movement. This means that there will be cases where a MH has moved to a new location, and has a new IP address, but binding update need not take place, even though Mobile IP would require it. We allow a level of "staleness" in the current location of the mobile host so that signaling traffic is reduced.

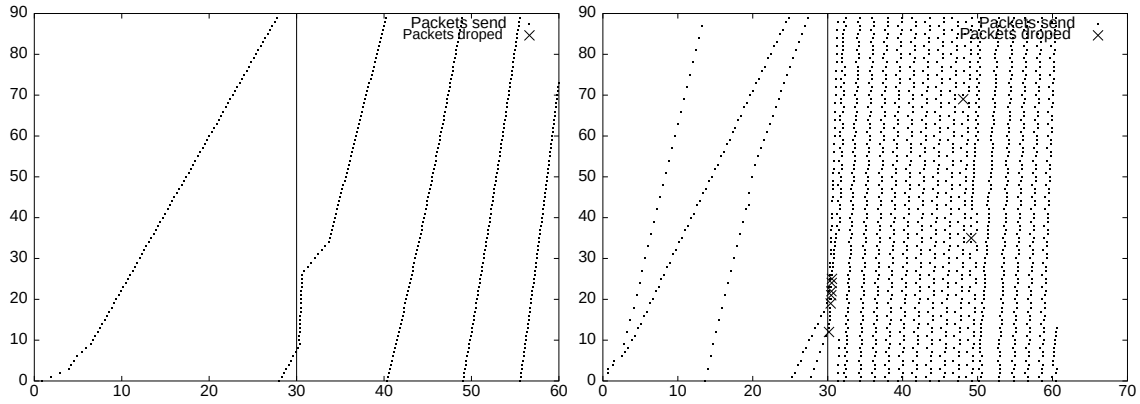
[1] Note that we are not concerned with the initial authorization process that the MH has to undergo.

soft_media_handoff()

1: **loop**
2: **if** new AP **then**
3: $dhcp\_MH : request()$
4: $sctp\_MH : add\_IP()$
5: **end if**
6: $sctp\_MH : send(CH, ASCONF)$
7: $hwnsp\_MH : send\_msg(CH, params)$ //proactive operation
8: **end loop**
9: $mobileip\_MH : send\_bu(HA, IP)$

Figure 48: The SCTP-based cross-layer media session handoff (MSH) algorithm.

However, when we allow a lag in the notification of the HA we must make sure binding updates sent to the CH, are delivered as soon as possible. These updates may happen while there is an ongoing data flow, and in some cases they can be done pro-actively.



(a) Packet sequence numbers for TFRC (b) Packet sequence numbers for MSH-SCTP

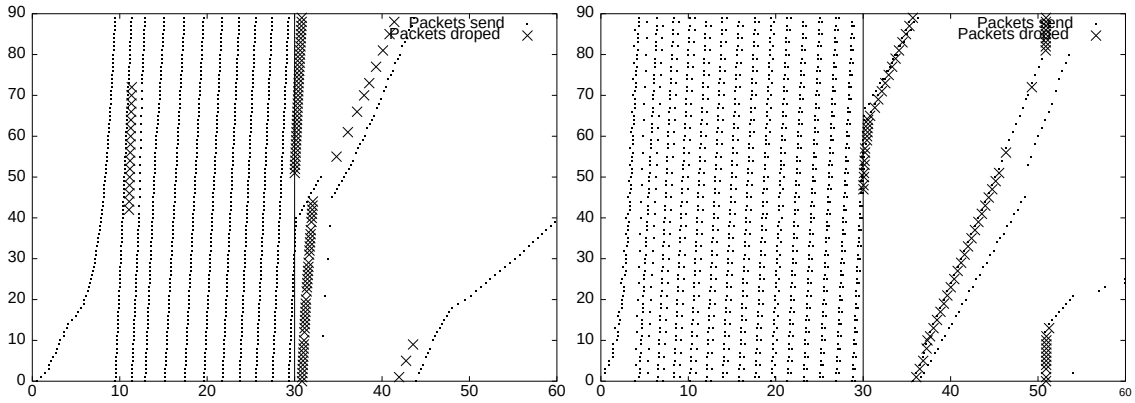
Figure 49: Vertical handoff from GPRS to UMTS link.

6.6.2 Simulations

A relatively different network setup was used for evaluating the performance of MSH-SCTP protocol in heterogeneous wireless networks. We used the ns-2 simulator [77] to evaluate the performance of the handoff algorithm under different transport protocols. According to the simulation scenario, handoff is performed at the 30th second from between two heterogeneous links. These links can be WLAN, GPRS, or UMTS. Link buffer size is setup to be 7 packets for GPRS and 20 packets for UMTS, the propagation delays 300 and 150

ms respectively, while the bitrates are 30Kbps for GPRS and 384Kbps for UMTS [42]. For a WLAN 802.11b network, the same parameters were set to 20 packets, 10 ms, and 6 Mbps respectively.

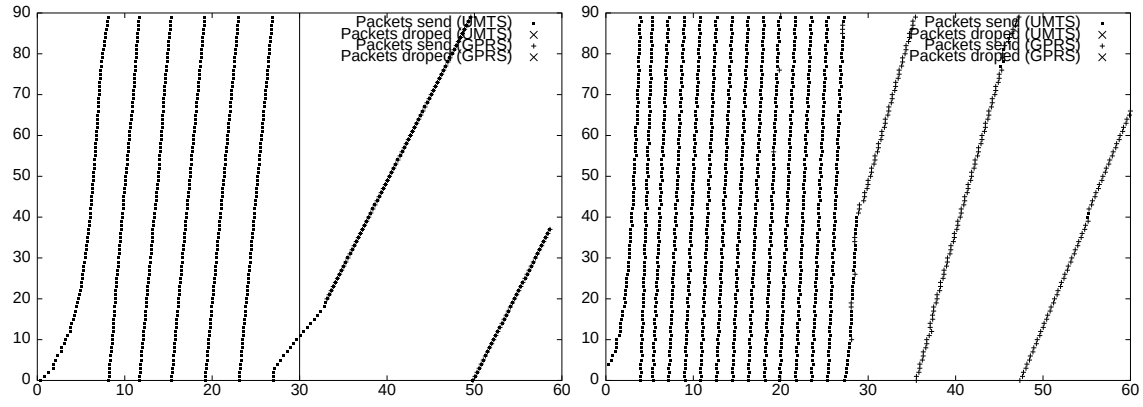
Figure 49 presents sequence numbers for the case of a vertical handoff from a GPRS to a UMTS links with TFRC, and MSH-SCTP. The MSH-SCTP protocol does not require any explicit handoff notification as recent solution for TFRC has employed [42]. The AIMD congestion control algorithm used by SCTP, adapts faster to the new link conditions as handoff from a GPRS link to a UMTS link takes place, and converges faster to the link bandwidth. Bandwidth utilization is increased especially during the critical time frame where the handoff takes place.



(a) Packet sequence numbers for TFRC (b) Packet sequence numbers for MSH-SCTP

Figure 50: Vertical handoff from UMTS to GPRS link.

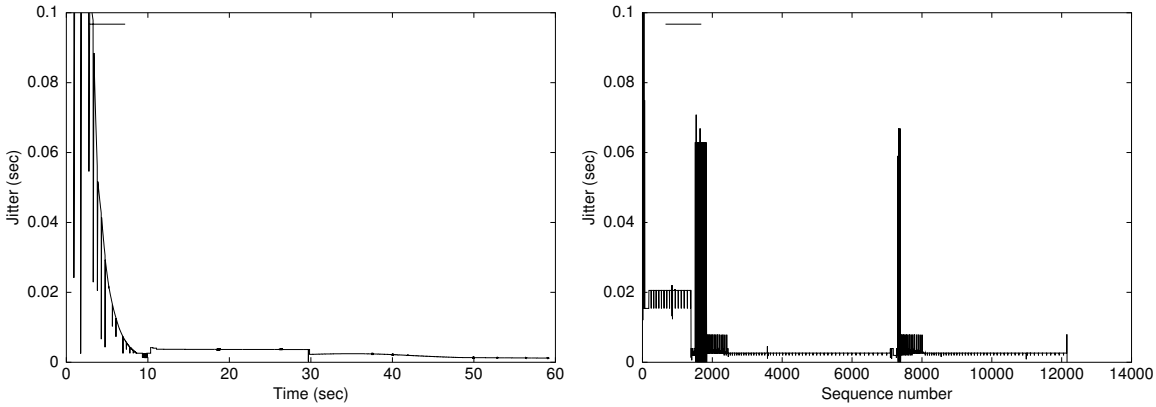
Now in figure 50, we present similar results but for a handoff from a fast UMTS link to a slower GPRS network. The situation in this case is different since TFRC cannot adapt fast to the new link resulting into a increased number of losses (figure 50(a)). However, when a window-based AIMD congestion control algorithm is employed, a reduced number of losses is observed during handoff, since the algorithm adapts fast to the new link and reduces the output rate very fast. A bunch of losses cause MSH-SCTP to reduce its rate sooner than TFRC. Now when a forwarding buffer is used during handoff, then TFRC can perform better since is can eliminate several packet losses (figure 51(a)). However, the adaptation to the new link still remains slow, due to the inherent behavior of the TFRC rate estimation



(a) Packet sequence numbers for TFRC (b) Packet sequence numbers for MSH-SCTP

Figure 51: Vertical handoff from UMTS to GPRS link with a forwarding buffer.

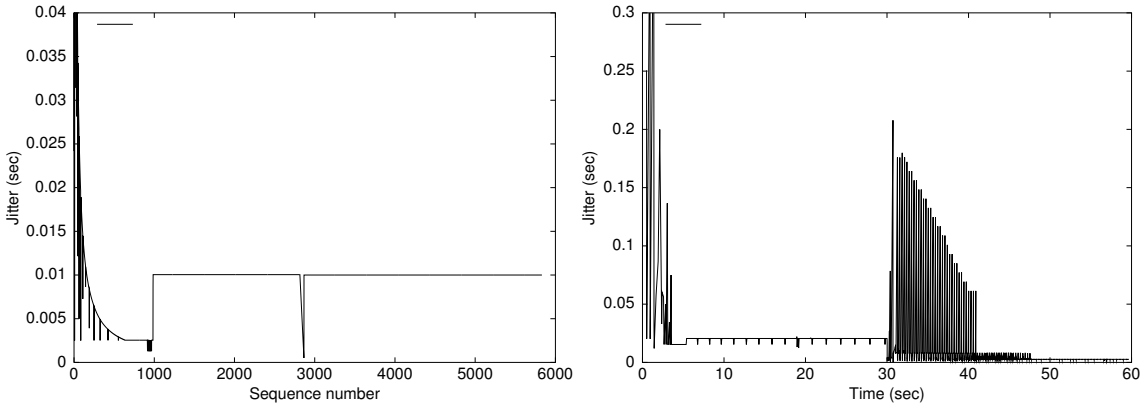
algorithm. When the MSH-SCTP protocol is used, we can see in figure 51(b) that packet losses are also eliminated and throughput is not throttled back. In this case we have the best combination for media flows where both packet losses are eliminated and fast rate adaptation is achieved.



(a) TFRC (b) MSH-SCTP

Figure 52: Jitter for handoff experiment with CBR flows from a UMTS to WLAN link.

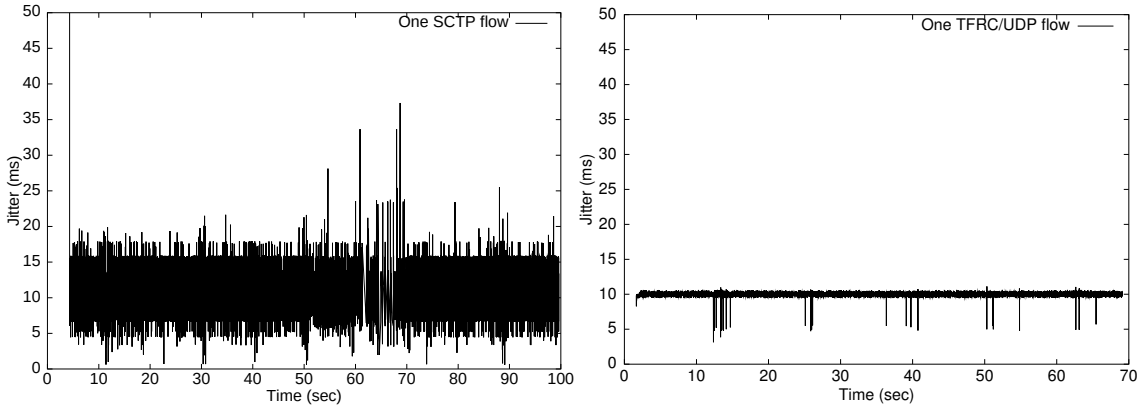
We also evaluated the performance of MSH-SCTP during handoffs in terms of the experienced jitter, since the TFRC protocol has as its primary objective stable jitter so that media flows can be efficiently supported. Figure 52 presents results for a UMTS to WLAN handoff experiment. Even though TFRC suffers generally from less jitter, the adaptation to the new link is slow (figure 52(b)). However, for SCTP during handoff the jitter instantaneously



(a) UMTS to GPRS handoff with TFRC (b) WLAN to UMTS handoff with MSH-SCTP

Figure 53: Jitter for handoff experiment with CBR flows.

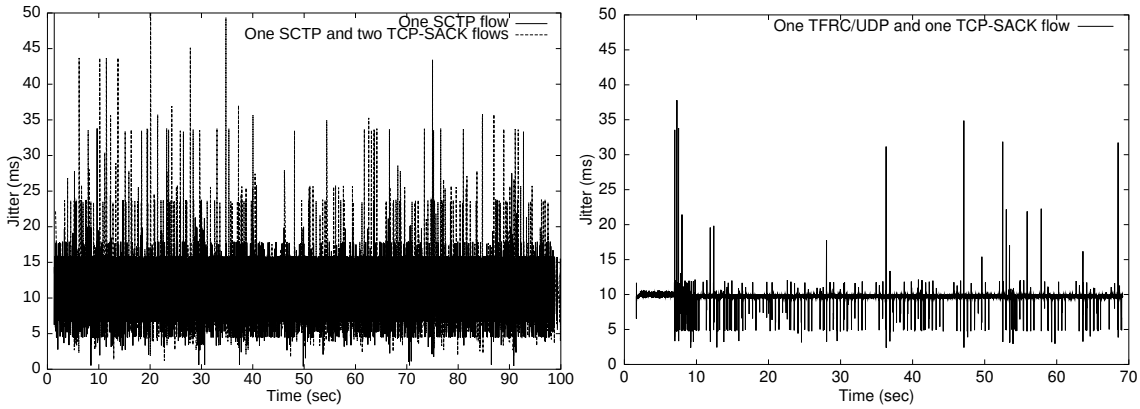
increases and drops fast when connection with the new link is achieved (figure 52(a)). Now, in figure 53(a) we present results for instantaneous jitter for a UMTS to GPRS handoff with the TFRC protocol. In this figure we see that TFRC can achieve pretty stable jitter despite the initial instability. For SCTP however, when a handoff happens from a fast WLAN to a slower UMTS, the protocol suffers from significant jitter for quite some time after the handoff instant.



(a) AIMD part of the hybrid algorithm (b) TFRC part of the hybrid algorithm

Figure 54: Jitter for one SCTP/VoIP flow in the 802.11b WLAN.

Figure 54 depicts the jitter when a single flow with combo-workload exists in the WLAN (as derived in chapter 3). More specifically, figure 54(a) shows the global behavior of the AIMD algorithm, while figure 54(b) depicts jitter of the CBR part of the flow. Figure 55



(a) AIMD part of the hybrid algorithm (b) TFRC part of the hybrid algorithm

Figure 55: Jitter for one SCTP/VoIP and 3 TCP flows in the 802.11b WLAN.

depicts results for the experienced jitter when three additional TCP flows exist in the WLAN. These results also verify that the TFRC microflow can maintain a stable jitter since it can use packet slots from the globally available AIMD window. Or in other words, it is utilizing the AIMD algorithm to probe for the wireless bandwidth and then the TFRC flow can commit the outgoing packets at a more stable rate.

6.7 Conclusions

In this chapter we developed an analytical-driven video streaming protocol, suitable for heterogeneous wireless networks where both handoffs and wireless errors are possible. Subsequently, we used our comprehensive model the development of protocol for managing an end-to-end video streaming session in a heterogeneous wireless environment. The proposed protocol, can be easily implemented on top of the aforementioned transport protocols. Performance evaluation revealed the protocol's ability to maintain low playback buffer underflow rate.

In the second part of this chapter we proposed a new media handoff protocol, which when it is combined with the previously developed streaming protocol can assure the best possible QoS for a media streaming session. With the development of this complete protocol suite, we demonstrate that the use of analytical, closed-form models that capture the effect of heterogeneous wireless networks, can be utilized by a practical cross-layer protocol that

controls the unicast streaming session. We prove with extensive experimental and simulation results that smooth media handoff can significantly improve performance by maintaining a more constant output rate.

CHAPTER VII

MULTIPATH TRANSPORT PROTOCOL MODELS FOR WIRELESS VIDEO STREAMING

In this chapter we are concerned with the use of TCP for multipath video streaming in wireless mobile networks. Our objective is to demonstrate that the use of analytical performance models for a hybrid wireless/wired network, can be used for driving the behavior of a multipath video streaming protocol so that the delivered video quality is improved. To achieve this objective, we initially develop a stochastic closed-form latency model that captures the behavior of TCP when multipath transport is considered. Based on the developed model, we devise a set algorithms for optimizing wireless video streaming. More specifically, we initially present an adaptive playback adaptation algorithm that operates only at the client without intervention of the streaming server. The second algorithm controls multipath scheduling of video packets, and can operate on top of any multipath transport protocol. Main task of this algorithm, is the estimation of expected latencies of video packets, and the proper allocation to the outgoing paths based on the playback deadlines at the client. Finally, we introduce the idea of multipath ARQ, and a new algorithm, that intelligently decides the allocation of video packet retransmissions to the available paths.

7.1 Introduction

Mobile wireless devices today are capable of using multiple wireless access technologies like WLAN, 3G cellular, or MAN (e.g. WiMax). Exploiting the full potential of these access networks, is of paramount importance when always on, high QoS services have to be provided to mobile users. Especially for media applications like pre-recorded or live video streaming, the use of multiple interfaces might prove crucial for meeting the goals of high throughput, minimized latency, low jitter, and service continuity in the next generation heterogeneous wireless networks.

In these networks, the highly popular TCP/IP Internet protocol suite will have to be used [38]. Of course, it is well known that TCP is considered unsuitable for video streaming applications. The main reasons are the rapid throughput fluctuations and the reliability mechanism which incurs additional delays [110]. Therefore, it is generally believed that the transport protocol of choice for video streaming should be UDP, on top of which several application specific mechanisms can be built [110]. However, the absence of congestion control from UDP can cause performance deterioration for TCP-based applications if wide-scale deployment takes place in the Internet [104, 52]. In addition, several commercial media applications like Quicktime [85], Windows Media [111], and Real Media [87], are already using TCP for video streaming, and haven proven the viability of such a solution.

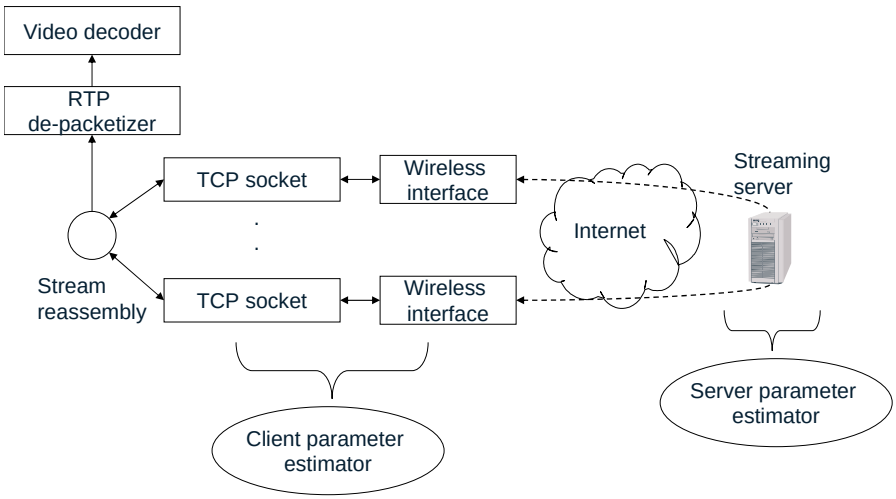


Figure 56: The proposed multipath streaming architecture based on TCP.

Therefore, in this chapter we will also assume the use of the popular TCP protocol, and we will explore how it can be used for a video streaming application, when multiple paths can be used concurrently. Figure 56 depicts a high level view of the proposed media streaming model based on TCP. The crucial difference of our work when compared with

the bulk of multipath streaming mechanisms, lies in the development of analytical models that capture the behavior of TCP. We follow this approach so that we can exploit the understanding of the protocol behavior and optimize video streaming. Based on the developed models, we study the performance of multipath streaming, when a transport layer protocol is controlling the packet path diversity process. We initially develop a client-driven playback adaptation algorithm, that estimates the expected latencies of video packets, and regulates the playback rate accordingly. Subsequently, we present a multipath scheduling algorithm, which is again analytically-driven and works in conjunction with the multipath transport models that we developed. Another novel idea that is introduced in this paper, is that of multipath retransmission, which means that the sender intelligently decides the retransmission allocation to available paths, based on the latency estimates for each path.

7.2 Related Work

The last few years a considerable amount of works in the area of multipath video streaming have been presented [7, 65, 105, 57]. The majority of them are based on the use of multiple description coding (MDC) techniques. Multiple description coders produce a number of self-contained encoded descriptions for a video sequence with lower quality than the single description equivalent of the initial sequence. Even if one description is received, the application will be able to reproduce the video sequence at a lower quality. However, as the number of the descriptions received at the decoder is increased, the display quality is improved [40]. One disadvantage is that the bitrate used for MDC is higher for achieving the same quality single layered compressed video [40]. When it comes to the network aspect, all the approaches for multipath streaming of MDC video, make assumptions about path dependence and the existence of network overlay infrastructures. Novel though, these systems are far from being employed over the current Internet [38]. In another work reported at [57], the authors compare multipath video streaming approaches at the application, transport and network layers without delving into the detailed algorithms of the protocols and their interactions. In a more recent works [25, 65], we see typical systems for multipath streaming of pre-encoded MDC video, where two paths are used for increased error resilience.

Finally, in [98] the authors proposed a system for multipath video streaming with TCP, but the existence of multiple paths is only exploited on the reverse path by duplicating acknowledgments.

7.3 TCP Latency Model for Multiple Asymmetric Paths

In this section we analyze the decomposed end-to-end path model, and explain the procedure we used for estimating all the crucial parameters that affect the delivery of a media stream for a specific transport protocol. Objective of the performance model, is the evaluation of the latency for the TCP or TFRC transport protocols. Subsequently, we incorporate into the end-to-end latency model the behavior of the playback buffer at the mobile client, since it is a central part in any media communications system. With this modular approach, we successively build a model that accommodates more and more parameters.

Figure 57 depicts the channel model where the transport protocol is using simultaneously $N = 2$ paths, while each path consists of a number of links. The wireless access network (AN) is modeled as a two-state Markov chain, and can be at any moment in one of two states, namely good (G), or bad wireless (W). Concerning the core networks (CN), they are modeled by using the Bernoulli path model, which means that two states exist, namely good (U) and bad (B). Therefore the aggregate packet loss probability for each path will depend on whether at least one channel is in bad state:

$$P = P(W, B) + P(G, B) + P(W, U) = 1 - P(G, U) \tag{105}$$

Obviously the same will hold for any additional path. Now, the transition probability matrices for each of these networks will be:

$$M_{AN} = \begin{pmatrix} \pi_{GG} & \pi_{GW} \\ \pi_{WG} & \pi_{WW} \end{pmatrix} \tag{106}$$

$$M_{CN} = \begin{pmatrix} \pi_{UU} & \pi_{UB} \\ \pi_{BU} & \pi_{BB} \end{pmatrix} \tag{107}$$

A key component for the development of a stochastic model, is the definition of the random variables of the problem, and their correlation. We consider that the round-trip times (RTT) of all the paths are i.i.d. random variables that are independent of the size of the congestion window [80]. Another assumption that we make is that since the losses do not originate from congestion at the same bottleneck, their respective RTTs are independent.

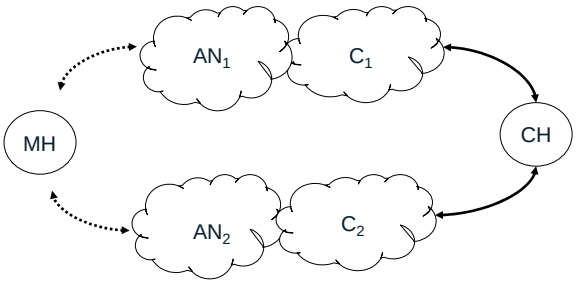


Figure 57: Channel model with multiple access and core networks.

7.3.1 Latency Model

An important concern when developing stochastic models is the validation of the stationarity assumption [89]. In our case, the stationarity assumption simplifies the calculation of the random variable that describes the latency, since both the mean and variance will have a constant value. We make this assumption, so that an analytically tractable model is obtained. Now the first step before we calculate the average value for the latency, is to estimate the packet loss rate in the end-to-end path.

Wireless packet loss rate (P_W): We start by the estimation of the packet loss rate due to wireless errors in the access networks. We adopt the Gilbert path model for capturing wireless channel behavior, since it is simple and fairly accurate [8]. Therefore, the packet loss probability due to wireless errors will be given by:

$$\pi_W = \frac{\pi_{GW}}{\pi_{GW} + \pi_{WG}} \tag{108}$$

The transition probabilities are calculated using maximum likelihood estimators [18]: $\hat{\pi}_{GW} = \frac{n_{GW}}{n_G}$, where n_{GW} is the number of times in the ACK messages that W state follows G state, n_{WG} is the number of times G follows W, and n_G is the number of times a good state is followed by a good. In a practical application if we want to find whether the wireless channel is in the bad wireless state, the mobile host can obtain it from the MAC layer.

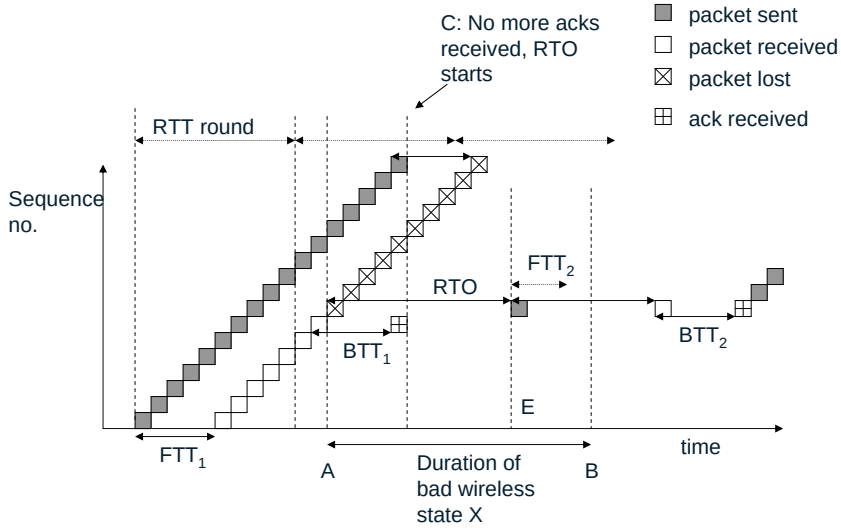


Figure 58: Packet-level TCP behavior at the sender during IP layer handoff.

Figure 58 will be used for explaining the behavior of TCP during a bad wireless state that lasts X seconds. The variables FTT_1/BTT_1, and FTT_2/BTT_2 describe the forward and backward trip times for the two access networks respectively. As we can see from this figure, at time instant t_A, layer 2 connection is lost and at time t_A+RTO, the TCP sender retransmits the first lost packet. Now if $t_B > t_C + FTT_1$, then clearly no duplicate

acknowledgments will be received at the sender, and the only way for TCP to resume the data flow is by expiration of the RTO of the first lost packet. On the other hand, as X shrinks, and if $t_B \leq t_C + FTT_1$ then the probability to receive a number of the last packets (close to time instant t_C) is increased. If this happens, then the sender would fast retransmit the first lost packet, resulting into a faster recovery. If we rewrite the previous equation we have (and because $t_C = t_A + BTT_1$): $t_B \leq t_C + FTT_1 \Rightarrow X \leq RTT_1$. Therefore, the packet loss probability, is the probability of that the one way latency L, is smaller than the duration X, i.e. $P_W = P[L < X]$. If we decompose the latency to the two components from which it consists, then the previous equation is written:

$$P_W = P[L_N + L_{protocol} < X] \tag{109}$$

The two variables L_N and $L_{protocol}$ express the latency induced by the network and the transport protocol respectively. We define as f_{L_N} the distribution of the end-to-end network induced latency. The value for X, is a parameter that depends on the wireless channel. In this paper, we assume that this random variable is exponentially distributed with mean T_H. Since there is a need to estimate the end-to-end latency, we will proceed to find it next.

Steady state latency distributions: The distribution of the TCP latency can be in the general case heavy tailed [22], since the use of the best-effort packet forwarding service that the core of the Internet supports, cannot provide any delivery guarantees. However, its precise p.d.f. will depend on the assumptions we make about the packet loss model. Let now R and Y denote the random variables of the RTT and the RTO respectively. From figure 58 we can see that when $L < X$ two more cases arise:

Case 1 ($X > t_E - t_A$): This condition means that the mobile host will still be in the bad state, while the RTO expires the first time (time instant t_E in figure 58). So if we want to find the probability that the AN will be in bad wireless state after time ϵ, this can be expressed as:

$$P\{S(t_E + \epsilon) = G | S(t_E) = W\} = P_g(\epsilon) \tag{110}$$

The term $P_g(Y)$ expresses the probability for the channel to be good state after time Y, when the RTO expires. The above holds due to the memoryless property of the distribution of X, causing thus the channel state at time plus ϵ, to be independent from the state at t_A. Therefore:

$$P_g(\epsilon) = P\{S(\epsilon) = G\} = \frac{T_G}{T_H + T_G}(1 - e^{-\epsilon/T_G}) \tag{111}$$

So the average TO duration, for every possible RTO value, will be equal to the probability that the channel is in good state at that specific RTO (given that it was bad before) times the value of the RTO. So this value will be:

$$L_{TO}(Y) = \sum_{i=1}^{6} 2^{i-1}Y \times P_g(iY) \prod_{j=0}^{i-1}(1 - P_g(jY)) + 64Y P_g(64Y) \prod_{i=0}^{6}(1 - P_g(iY))$$

The product term in the previous equation expresses the probability that the AN was in bad wireless state, when the TO expired in the previous time instants before i. In addition, after the first six consecutive TOs the value of the TO will be fixed to $64Y$ [99]. The last term on the above equation captures this effect.

Case 2 ($X < t_E - t_A$): In this case, we can see from figure 58, that it will also be $t_C < t_E$. This means that the sender will not experience a TO, but instead it will receive three duplicate acknowledgements (TD), and so it will fast retransmit the first missing packet. The average latency introduced due to this event is:

$$L_{TD} = \int_{t=0}^{T_H} t P_g(t) dt \tag{112}$$

Therefore, the total latency for TCP can be expressed from the previous two equations:

$$L_{TCP} = \sum_{l=0}^{\infty} P[l < L] = \int_{t=Y}^{T_H} L_{TO}(t) f_X(t) dt + \int_{t=0}^{T_H} t P_g(t) dt \tag{113}$$

The first term in the above equation follows from the assumption that the disruption time due to handoff X, is exponentially distributed with a mean equal to T_H and a p.d.f $f_X(t)$.

Concerning the core network induced delay L_N, that occurs mainly due to buffering at the routing infrastructure, it has been shown that it could be modeled as a shifted Gamma

distribution [25, 64]:

$$f_{L_N}(t) = \begin{cases} \frac{\lambda e^{-\lambda t}(\lambda t)^{\nu-1}}{(\nu-1)!} & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \quad (114)$$

We will follow this distribution for the latency in this chapter, so that the derivation of an analytical closed form solution is possible. Several possible analyses can be preformed in order to model more accurate the core network performance, but this research is out of the scope of this chapter.

7.3.2 Playback Buffer Model

In the previous subsection we estimated the average expected latency L for the TCP packets, and the packet loss rate P_W. These two quantities are primarily related to the duration of the bad wireless state X. However, when we consider the playback buffer at the client, we need to calculate one more parameter. This is the probability of a packet missing its deadline when the playback buffer exists. Calculating this value, will eventually help estimate the expected number of buffer underflow events.

So if we consider the existence of the playback buffer, the probability for a packet to be delayed, and miss its deadline, can be formulated as:

$$\begin{aligned} P_D &= P[\text{sent time} + \text{latency} + \frac{\text{buffer occupancy}}{\text{playback rate}} \leq \text{deadline}] \\ P_D &= P[t_s + L_N + L_{protocol} + \frac{b_1}{c_1} \leq t_d] \end{aligned} \quad (115)$$

In the above equation t_s is the time that a video packet was sent, while t_d is the playback deadline for this video packet. For TCP and TFRC this expression will be different. TFRC does not introduce any latency since it does not control retransmissions and so $L_{TFRC} = 0$. In the case of TCP, L_{TCP} will be the latency incurred by the TCP retransmission mechanisms, due to the packet loss rate P_N in the core network. Therefore equation 115 will be:

$$P_D^{TCP} = P[t_s + L_N + P_N L_{TCP} + \frac{b_1}{c_1} \leq t_d] \quad (116)$$

$$P_D^{TFRC} = P[t_s + L_N + \frac{b_1}{c_1} \leq t_d] \quad (117)$$

These values will later help us quantify the model performance in terms of correct deadline estimation at the sender. By elaborating on the previous formulas we can write for TCP:

$$\begin{aligned} P_D^{TCP} &= P[L_N + P_N L^{TCP} \leq a] \\ &= \int_{-\infty}^{+\infty} F_{L_N}(a-y) f_{L^{TCP}}(y) dy \end{aligned} \quad (118)$$

with $a = t_d - t_s - \frac{b_1}{c_1}$. The distributions in these formulas are simple sums of exponential values, making thus the derivation of a final solution straightforward.

7.4 Multipath Video Streaming and Adaptive Playback

Throughout the next sections, we will see how we can utilize the previously developed models for the design of practical adaptive video streaming algorithms. Our central idea is that a better understanding of the protocol behavior, can be used so that we adapt several functions of the media application.

Based on the previously derived equations, we can define the optimal playback rate c given the current playback buffer occupancy b. In order for the playback buffer not to underflow the duration of the available media for playback must exceed the latency of the next set of arriving packets L. Or this can be written as: $\frac{b}{c} > L_j$ with j $\in$ Paths.

So based on our analysis till now, we can devise a multipath playback adaptation algorithm that can be seen in figure 59. We use the term video data unit (VDU), to describe a single video packet. This algorithm operates at the client and it basically calculates the necessary model parameters based on the packet delivery events (lines 2-4). The packets are then classified according to whether they missed their playback deadline or not and this process helps refine the estimates for the the two values P_D and L (lines 5-9). Subsequently, the playback rate c, is adapted according to equation **??**, so that fluctuations during the

```
adaptive_mpath_playback()
 1: for all VDU received do
 2:   t_{r_i^j} ← // recorded arrival time for VDU_i at path j
 3:   P̃_D^j ← // estimate P_D for path j
 4:   L̃_H^j ← // estimate L for path j
 5:   for all j ∈ Paths do
 6:     if t_{r_i^j} < t_{d_i} then
 7:       add_buffer(VDU_i) // playback OK
 8:     else
 9:       record missed t_{d_i} // playback missed
10:     end if
11:   end for
12:   play(VDUs)
13:   b_1 ← recalc(size)
14:   c_1 ← recalc(P̃_D^j, L̃_H^j)
15: end for
```

Figure 59: Proposed playback adaptation algorithm for multipath transmission with TCP.

delivery of media packets can be accommodated. Note that during this process is performed individually for each one of the joint wireline/wireless paths that are in use.

In the next subsection we will evaluate whether this approach can minimize two important quantities which are the number of buffer underflow events and the initial preroll delay Δ.

7.4.1 Experimental Setup

The network testbed for our experiments consists of a client/server configuration that are linux boxes while the middlebox is freeBSD machine that acted as a router and emulated the multiple paths. We used the middlebox with the Dummynet software [36], for emulating the wireless packet losses in the access networks, and buffer overflows in the core network routers. The sequences FOREMAN, AKIYO, and COASTGURAD [94], were used for the video streaming experiments. The H.263 encoder [44] was used for encoding the YUV sequences into various bitrates. The video units were packetized into RTP packets and the sent to the transport protocol which is our case were TCP and UDP/TFRC. Due to the short duration of the sequences (150 frames), they were repeatedly fed as input to the encoder, so that encoded sequences of longer duration could be obtained. The capacity of

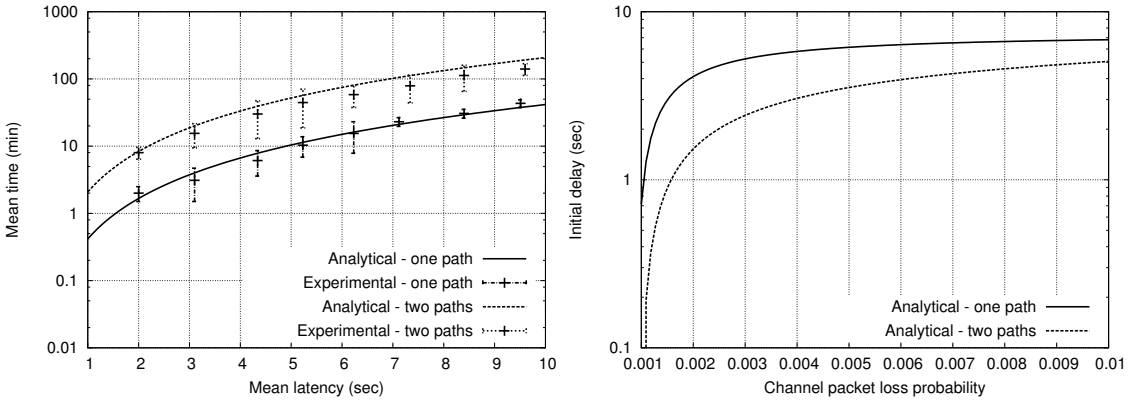
the bottleneck link between the two routers was set to 250Kbps. All the sequences were encoded at 128Kbps with a target frame rate of 15 fps. The results were obtained by running the same scenario 100 times and averaging the PSNR values of the same experiments. A summary of all the parameters used throughout our experiments in this section, can be seen in table 5.

Table 5: Model parameters used for the multipath video streaming experiments.

Network Parameters		Protocol parameters	
$AN : T_H$	(WLAN) 50 ms	T_0	200 ms
$AN : T_G$	(WLAN) 1000 ms	W_{max}	6 MByte
$AN : T_B$	(WLAN) 100 ms	W_0	1 segment
$CN : T_G$	1000 ms	MSS	1460 bytes
$CN : T_B$	5 ms	Mobile speed	1-20 m/s

7.4.2 Experiments

In this subsection we present a set of experiments for TCP-based multipath video streaming with the wireless network configuration that we showed in figure 56. This part of the experiments has as primary objective to validate whether the playback algorithm shown in figure 59, can reduce the number of buffer underflow events.



(a) Mean time between underflow events for BUP<0.01

(b) Initial preroll delay needed to achieve a BUP<0.01

Figure 60: Analytical and experimental results for the multipath playback adaptation algorithm with TCP. TCP parameters: $RTO_0 = 200$ ms, $MSS = 1460$ bytes, $W_0 = 1$ segment, $W_{max} = 6$ MB, video duration is 100 sec.

Experimental results compare the proposed model-based algorithm with the most closely related work identified in the literature. We reproduced the experiments reported at [102], where the authors present a playback adaptation algorithm for wireless channels. A very interesting result is presented in figure 60(a). It depicts the mean time between two buffer underflow events as a function of the end-to-end latency between the two endpoints. We see that when two paths were used, a reduced number of buffer underflow events when compared to a single path.

Now figure 60(b) depicts the initial delay needed at the playback buffer, in order to achieve a buffer underflow probability (BUP) less than 1%, as a function of the packet loss rate for TCP. It is obvious that the value of the initial delay is increasing very fast as the packet loss probability is increased until it saturates in a value close to 8 seconds for a video stream that has duration of 100 seconds, when the proposed TCP model is used. However, a streaming application that is using TCP and attempts to simply estimate the network induced latency, has as a result a higher rate of buffer underflow events early in the video streaming session. In order to overcome these underflow events, a larger initial delay Δ can be selected, by using estimate of the RTT [104].

7.5 Analytically-Driven Multipath Video Scheduling Algorithm

In this section we extend the problem of optimal video streaming one step further: Given the encoder buffer which contains a set of video packets Φ, that belong to frame f (for F possible frames), how can we transmit these packets so that their arrival time is minimized? For a given set of allowed schedules S, the answer to this question can be formally written as:

$$s = arg \min_{s \in S} E[latency(f, \Phi, t_d^n)] \tag{119}$$

where n is the number of the video packet that belongs to group Φ, and t_d^n is the playback deadline for this packet. The search space of the possible schedules S, depends on the number of video packets that each TCP pipe of the multipath streaming system can send

as a chunk. The question that is raised now is how to formulate an algorithm for this problem.

server_ppd_schedule()
1: $G \leftarrow$ // estimate TCP goodput
2: $\tilde{r}_i^j \leftarrow$ // estimate arrival for VDU_i^n at path j
3: $\tilde{P}_D \leftarrow$ // estimate P_D
4: **for all** VDU $i \in \Phi$ **do**
5: **for all** $i \in$Paths **do**
6: **if** $\tilde{t}_{r_i^j} < t_{d_i}$ **then**
7: Add VDU_i in $\Phi_j^{\#}$
8: **else**
9: Drop VDU_i
10: **end if**
11: **end for**
12: **end for**
13: **for all** $VDUs \in \Phi_j^{\#}$ **do**
14: Find optimal set Φ_j^* that min(E[D])
15: $set\_priority(send\_buffer\_j, MB_i)$
16: **end for**
17: send(MBs$\in \Phi_j^*$,Path j)
recv_report()
1: Client may report in feedback messages both $\tilde{e} = |t_r - \tilde{t}_r|$ and G.

Figure 61: Proposed multipath scheduling algorithm with transport protocol awareness.

Our intuition says that at the streaming server, we should transmit a video packet to the path that will allow the playback deadline to be met sooner. We devised an algorithm that implements the above observation, and can be seen in figure 61. This is what lines 4-9 of the algorithm are exactly doing by identifying the set of video packets that will actually reach the decoder in time based on the model prediction. Essentially this part of the algorithm estimates the expected delivery time of data packets over all the available outgoing paths. After this decision is made the algorithm invokes the greedy scheduling algorithm which identifies the optimal schedule according to the estimated path latencies (lines 11-14). An optimal schedule $S_j^{\#}$ is identified for each path j. Now the relative priority of each packet is enforced through a prioritization mechanism so that video packets that have an earlier deadline, are committed to the fastest TCP path first. We also depict in the last line of the algorithm, a line that indicates the ability of the client to provide feedback to the

sender. The feedback may involve among others, the actual reception time for a packet and the actual goodput, allowing thus the implementation of more efficient estimators at the sender.

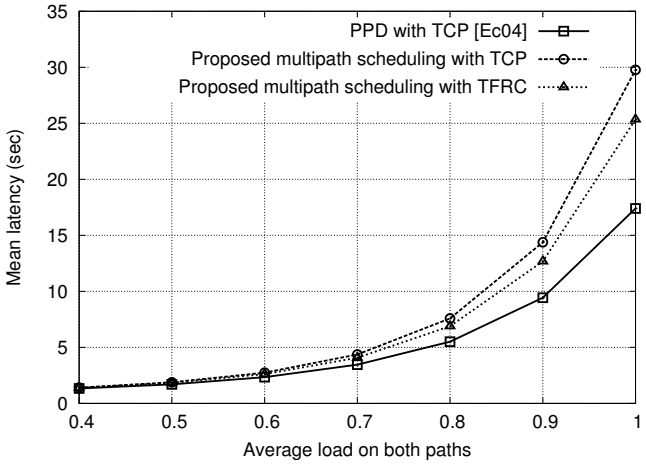


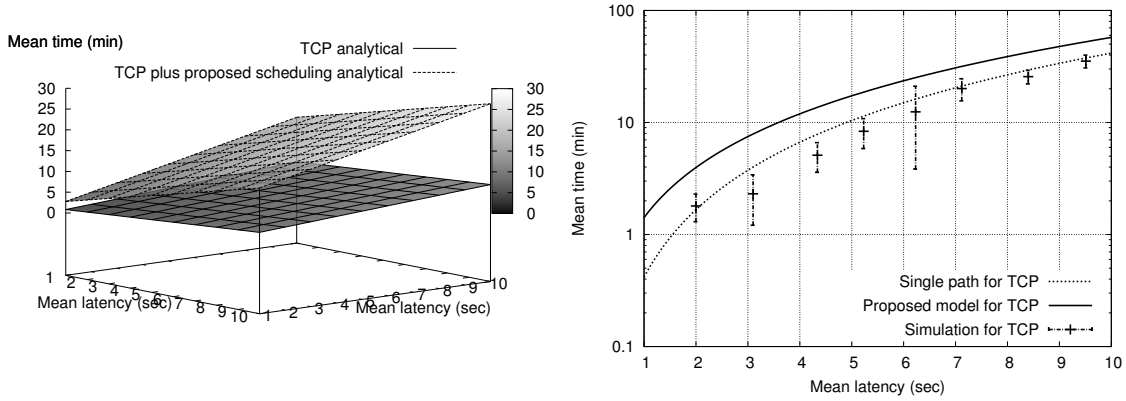
Figure 62: Throughput as a function of the load on two concurrently used paths. Parameters: $RTT = 500$ ms, $plr = 0.02$, $RTO_0 = 200$ ms, MSS=1460 bytes, $W_0 = 1$ segment, $W_{max} = 4$ MB

The crucial difference of the proposed multipath media scheduling algorithm when compared with related work, is that it takes into consideration the inner dynamics and operation of the transport protocol. The majority of the related work usually operate under the assumption that the underlying protocol is always UDP, and no congestion control performed. We will show with our experimental results, that the use of analytical video streaming algorithm can have significant effect on the delivered video quality for wireless mobile networks.

7.5.1 Experiments

Figure 62 presents the average latency for the transport of a 1MB media clip, as a function of the load on the two paths. The results presented are the average values 100 simulation runs. We compare our scheme with a packet path diversity (PPD) methodology reported at [98], where the authors also assume the concurrent use of two paths with TCP, and acknowledgments that are send on both of the reverse paths. The results for the proposed multipath scheduling algorithm, depict an improvement in the average throughput for the streaming algorithm, as figure 62 indicates. One of the reasons is that the media-oriented

methodology developed at [98], simply duplicates the packets and the acknowledgments on both paths resulting into wasted bandwidth resources. However with our mechanism, different video packets are distributed to the available paths, and error recovery is realized through the retransmission algorithm that TCP inherently supports. This essentially means, that redundancy is essentially introduced when it is need, and this is the case only when a packet loss happens and a retransmission must take place. In figure 62, we also present similar results but when TFRC was used instead of TCP. This protocol can similarly achieve better performance when compared with the single path streaming mechanism.



(a) Numerical results for MTBBU as a function of the latency in the two used paths

(b) Simulation results for MTBBU and target BUP≤0.01 for TCP and TFRC

Figure 63: Mean time between buffer underflow (MTBBU) events for multipath media transport with TCP. WLAN parameters: link $buffer\_size = 20$ pkts, $link\_delay = 10$ ms, $bitrate = 6$ Mbps.

The next experiment involved the evaluation of the multipath scheduling algorithm for the transport of real-time media data. The experimental setup was similar with before, only that this time a 256Kbps CBR encoded sequence was used instead of an infinite data backlog as before. Figure 63(a) depicts the mean time between buffer underflow events as a function of the latency of two paths that are used concurrently. While TFRC can perform better when compared with TCP, the multipath scheduling algorithm can monitor precisely the packet delivery and achieve a reduced number of buffer underflow events. Figure 63(b) depicts simulation results that show the normalized rate of buffer underflow events, similar to previously presented results. Multipath scheduling results into significant decrease on the number of buffer underflow events mainly because of the utilization of a

faster path that deliver video packets sooner. While the use of just one path may not be able to assure constant data flow, the use of the second path can achieve that, which is the main concern especially for media applications. We see that the possibility of using two paths concurrently, and due to the assumption of independent paths, this approach can significantly improve the perceived video quality. We can also observe in the same figure that the experimental results follow closely the analytically derived model estimates. We observed that the margin of error was less that 10% for the mean time between buffer underflow events.

7.5.2 Comparison with MDC

In this subsection we will present comparative results of the proposed multipath scheduling scheme with multiple description coding (MDC) path diversity mechanisms. For the MDC experiments, we encoded two sequences into a base and enhancement layer 128Kbps and 64Kbps respectively. For the MDC experiments, we reproduced experiments reported at [25]. A short set of experimental results can be seen in figure 64. At higher packet loss rates, the MDC system provides higher quality due to the redundancy introduced by the two description. However, the proposed scheduling algorithm performs better for lower packet loss rates which are less than 5%. In this case, bandwidth utilization is considerably improved, since the level of redundancy that MDC introduces is not actually needed.

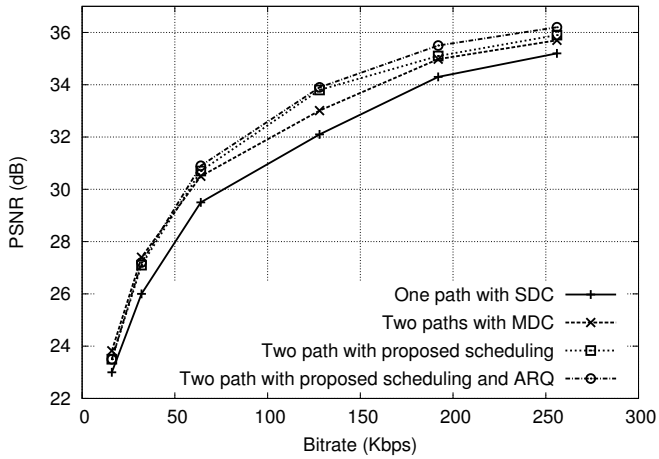


Figure 64: Distortion-rate performance results for QCIF AKIYO.

However, these results do not represent the final step of our analytically-based algorithm design procedure. In the next section, we will develop a multipath ARQ algorithm that can improve performance of the proposed multipath scheduling algorithm. The objective will be to improve error resiliency in the face of increased packet loss rate that takes place in one of the used paths.

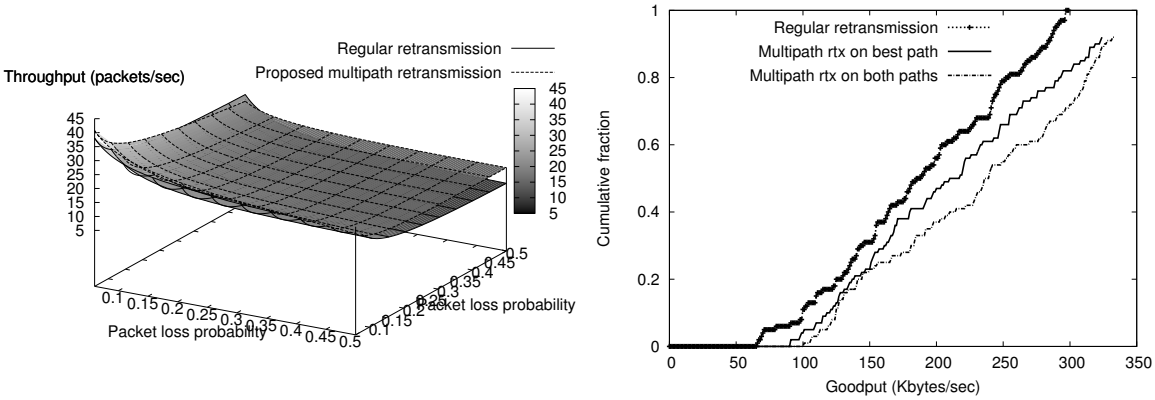
7.6 Multipath ARQ of Video Packets

Following up on the methodology that we have developed till now, we will present a new algorithm for implementing more efficient automatic repeat request (ARQ), when multipath video streaming is the target application. Consider again our TCP-based system model shown in figure 56, with a session that consists of N separate end-to-end TCP connections. Since TCP is our base protocol, the application cannot control retransmission of specific packets. However, what we see here is the potential to control retransmissions in a multipath video streaming setting. What we propose in this section, is to retransmit lost packets to a different path from the one originally sent, based on the knowledge we have about the estimated latency for each path. The intuition is that we want to retransmit packets from a fast path since they have already suffered significant delay and the playback deadline might not be met. Essentially, it is up to the streaming protocol design to define how and when it will commit a specific packet to a specific path. However, if a packet is sent through a specific TCP path, the application cannot enforce a re-assignment of a possible retransmission to another TCP pipe that corresponds to a different path. In order to solve this problem, the solution we devised is to replicate the packet that is of interest, to another TCP pipe. In this way the packet that is of interest to us will reach the receiver sooner.

7.6.1 Experiments

To validate our intuition behind the design of the multipath retransmission algorithm (MRTX), we performed a set of experiments that concurrently use two paths for a single media flow. Figure 65(a) depicts analytical results that indicate an increased packet loss probability for a specific path, which creates a significant burden to the other paths because of the retransmissions. This validates our intuition for the necessity of a careful

retransmission policy in a multipath transport protocol. Especially if a wireless path is involved, then the packet loss rate due to wireless errors will probably be higher, making thus the proposed algorithm even more useful.



(a) Aggregate throughput with MRTX for two paths(b) Cumulative fraction of the throughput with TCP with varying packet loss rates

Figure 65: Analytical and simulation results for the multipath retransmission algorithm.

Another issue that arises is that in this way the receiver may get the retransmitted packet from another end-to-end path to which of course it has to reply with an acknowledgment. The receiver will send the acknowledgment back to the same path or the source from where it received it. Implicitly this creates a path monitoring mechanism and updates the status variables of a specific path. Experimental results are depicted in figure 65(b). More specifically we present the cumulative fraction of data received for three different retransmission policies, namely selective retransmission on just one path, retransmission on both paths, and finally regular retransmission. The improved throughput observed, is attributed to the faster reception of the retransmitted packet, and consequently the faster acknowledgment of that packet. This results into faster turnaround times and higher throughput.

7.7 Conclusions

In this chapter we demonstrated that the use of analytical performance models can be used for driving the behavior of multipath video streaming protocols, so that the delivered video quality is improved. To achieve this objective, we initially developed a stochastic

closed-form latency model, that captures the behavior of TCP when multipath transport is considered. Based on the previously developed models, we proposed three algorithms for optimizing video streaming. More specifically,we initially presented an adaptive playback adaption algorithm that operates only at the client without intervention of the sender. The second algorithm controls multipath scheduling of video packets, and can operate on top of any multipath transport protocol. Main task of this algorithm, is the estimation of expected latencies of video packets, and the proper allocation to the outgoing paths, based on the playback deadlines at the client. Finally, we introduce the idea of multipath retransmission, and a new algorithm, that intelligently decides the allocation of video packet retransmissions to the available paths. Results for the developed algorithms are very promising, since we observed throughput improvement of nearly 10% over a typical multipath retransmission policy.

CHAPTER VIII

CONCLUSIONS

In this dissertation we addressed the problem of efficient media delivery in heterogeneous wireless networks. We developed a set of comprehensive models that characterized the performance of TCP in a variety of wireless mobile scenarios. Subsequently, we presented the design of algorithms that optimize wireless multimedia delivery by carefully considering the predictions of the developed models. The conclusions and important contributions from this dissertation are summarized below:

Transport Protocol Models for CBR and VBR Workloads: In this chapter we presented analytical models that characterize TCP and TFRC throughput for different traffic workloads, namely CBR, VBR and bulk traffic in a wireless/wired network setup. The first important conclusion that we draw from the analysis in this chapter, is that the assumption of flows with an infinite data backlog, may significantly affect the TCP throughput estimate in case of CBR and VBR workloads. We demonstrated that with our model, these predictions can be more accurate, leading to a better understanding of the protocol and workload interactions. Therefore, when performance is evaluated, someone should try to correlate carefully the transport protocol in use, with the actual workload. We identified TFRC's inability to provide high throughput service when the traffic workload is characterized by large rate variations (e.g. VBR). This means that a number of additional factors have to be considered before deploying the protocol for a media application. For wireless scenarios, the proposed model does not differentiate significantly. Even, the asymmetry in the packet loss probability across the wireline wireless networks does not significantly the throughput since both TCP and TFRC experience the aggregate packet loss.

Rate-Distortion Optimized Unicast Video Streaming with TCP: In this chapter we presented an analytical study that characterizes the performance of video streaming with the transmission control protocol (TCP). Initially, we developed an analytical model of the

expected video distortion at the decoder with respect to the TCP parameters, channel state, and error concealment method at the receiver. Based on this model we propose an algorithm for RD optimized mode selection (RDOMS) for video streaming with TCP. Experimental results for real-time video streaming showed PSNR improvement of nearly two db over currently proposed TCP-based streaming mechanisms. The next contribution is the development of a joint model of the TCP protocol, and the playback buffer at the receiver. Based on this model, we derived the optimal playback rate at the decoder. Subsequently, based on the two models, we proposed an algorithm, for RD optimized packet scheduling with TCP. Our results show an additional improvement of nearly one db, when packet scheduling is applied together with the RDOMS algorithm. Therefore, we demonstarted that TCP presents a viable solution for video streaming applications. Moreover, we showed that if additional optimizations can be preformed at the video encoder side, further quality improvement can be observed. The wide-scale deployment of TCP, and the ability to realize the proposed algorithms at the application level, can assure that the proposed streaming mechanism presents a viable solution.

Modeling the Effect of Handoffs on Transport Protocol Performance: In this chapter we presented a model for studying the effects of wireless handoffs in two transport protocols, namely TCP and TFRC. The model was found to be accurate for TCP in both the cases where HMIP and MIP-RO were used as the underlying mobility management protocol. However, the TFRC model predicts the expected throughput with even better accuracy, due to the simpler protocol algorithms. For example the worst case error for the TCP model was nearly 22% while for the TFRC model it was 13%. An important observation from the conducted experiments is that the use of buffering in the old access network, can significantly improved the delivered throughput. If the requirements of the system specify that no packet loss should take place, the rule of thumb for TCP, is that the buffer size should be equal to the bandwidth-delay product of the old access path times the expected disruption time. Concerning TFRC, we found that the required buffer size should be surprisingly bigger by 60% than TCP. The reason for that is the slow responsiveness of TFRC which does not drastically cut its rate, resulting in the need of a larger buffer.

We also introduced in this chapter the notion of the "recovery period", which is defined as the time required for the transport protocol to achieve the nominal throughput in a new link, after a handoff that lasted t_h. The slow-responsive rate control algorithm of TFRC, requires less time in order to recover when compared with TCP. However, we found that as the disruption time is increased, TFRC suffers from more packet losses than TCP, due to the slow-responsive algorithm, which is persistent on sending new packets to the network.

Video Streaming in Heterogeneous Mobile Wireless Networks: In this chapter we developed an analytical-driven video streaming protocol, suitable for heterogeneous wireless networks where both handoffs and wireless errors are possible. Subsequently, we used our comprehensive model the development of protocol for managing an end-to-end video streaming session in a heterogeneous wireless environment. The proposed protocol, can be easily implemented on top of the aforementioned transport protocols. Performance evaluation revealed the protocol's ability to maintain low playback buffer underflow rate.

In the second part of this chapter we proposes a new media handoff protocol, which when it is combined with the previously developed streaming protocol can assure the best possible QoS for a media streaming session. With the development of this complete protocol suite, we demonstrate that the use of analytical, closed-form models that capture the effect of heterogeneous wireless networks, can be utilized by a practical cross-layer protocol that controls the unicast streaming session. We prove with extensive experimental and simulation results that smooth media handoff can significantly improve performance by maintaining a more constant output rate.

Multipath Transport Protocol Models for Wireless Video Streaming: In this chapter we demonstrated that the use of analytical performance models can be used for driving the behavior of a multipath video streaming protocols, so that the delivered video quality is improved. To achieve this objective, we initially developed a stochastic closed-form latency model, that captures the behavior of TCP when multipath transport is considered.

Based on the previously developed models, we proposed three algorithms for optimizing video streaming. More specifically,we initially present an adaptive playback adaption algorithm that operates only at the client without intervention of the sender. The second

algorithm controls multipath scheduling of video packets, and can operate on top of any of the existing multipath transport protocols. Main task of this algorithm, is the estimation of expected latencies of video packets, and the proper allocation to the outgoing paths, based on the playback deadlines at the client. Finally, we introduce the idea of multipath retransmission, and a new algorithm, that intelligently decides the allocation of video packet retransmissions to the available paths. Results for the developed algorithms are very promising, since we observed throughput improvement of nearly % over a typical multipath retransmission policy.

APPENDIX A

TCP RECEIVER MODEL

In this section we present the development of a simple model, in the same spirit as before, for the goodput of a TCP receiver.

If we consider a path that is characterized by packet loss probability p, the probability to receive correctly up to l packets before a loss occurs, will be described by a Bernoulli trial process and it will be equal to $P[l = k] = (1 - p)^k p$. After a packet loss occurs, the receiver will start sending duplicate acknowledgments for every received packet. However, it will still be receiving packets from the sender until the sender is informed for the packet loss with three duplicate ACKs. Note that after a loss event, a number of $\beta_i = W_i - l_i - 1$ more packets will be send, where W_i is the value of the congestion window in the current RTT, and α is the total number of packets send correctly in the NL round. Given that b is the number of packets acknowledged with a single ACK packet, then the correlation between the duration of NL round and window size will be [80]:

$$\begin{aligned} W_i &= W_{i-1}/2 + X_i/b - 1 \Rightarrow \\ E[X] &= \frac{b}{2}E[W] = \frac{2+b}{6} + \sqrt{\frac{2b(1-p)}{3p} + \left(\frac{2+b}{6}\right)^2} \end{aligned} \tag{120}$$

The average number of packets that will be correctly received, assuming that all the packets after the one lost in the same NL round are also lost, will be $R = a + 2\beta$. However, we want to be able to calculate the average number of packets received during each RTT round, given that the duration of an NL round will correspond to $X_i + 1$ RTTs. If we look into equation 120 more carefully, we can see that in each successive RTT k of an NL round i, the sender transmits $\frac{X_i W_{i-1}}{2} + kb$ packets. This indicates that until there is a packet loss, a steady flow of at least $\frac{X_i W_{i-1}}{2}$ and $\frac{W_{i-1}}{2}$ packets will be sent during the entire NL round and each RTT respectively. Therefore, the total number of packets received in a round of

$X_i + 1$ RTTs will be:

$$\frac{X_i W_{i-1}}{2} + \sum_{0}^{X_i/b-1} kb + 2 \times \beta_i$$

So, the average number of the received packets per RTT round which is $r = a + 2\beta$, after a few algebraic manipulations becomes:

$$E[R] = 2\Big(\frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \Big(\frac{2+b}{3b}\Big)^2}\Big) - \frac{1}{p} - 2 \tag{121}$$

So the closed-form equation for the receiver goodput will be:

$$G^{BULK} = \frac{E[R]}{(X+1)RTT} \tag{122}$$

One interesting quantity is the goodput variance at the receiver, which is given from $Var(R) = E[R^2] - E[R]^2$. It is obvious that the problem is essentially reduces to the calculation of $E[R^2]$. We can easily find that $E[l^2] = \sum_{k=0}^{\infty} k^2(1-p)^{k-1}p = \frac{1}{p^2}$. Furthermore, the number of packets that were sent until the packet that was lost will also be equal to:

$$l_i = \sum_{k=0}^{X_i/b-1} (\frac{W_{i-1}}{2} + k)b \tag{123}$$

From the above equation, we can obtain the expected value of $E[\alpha]$, which when it is combined with equation 120 we have:

$$E[X^2] \;=\; 2bE[l] - bE[X]E[W] + bE[X] \tag{124}$$

By further manipulations of equation 120, and the use of equation 124 we get:

$$\begin{aligned} W_i^2 &= \frac{W_{i-1}^2}{4} + \frac{X_i^2}{b^2} + W_{i-1}\frac{X_i}{b} \Rightarrow \\ E[W^2] &= (2E[l] + E[X])/b \end{aligned} \tag{125}$$

Concerning $E[R^2]$ calculation, from equations 121 and 125, and since $\beta_i = W_i - l_i - 1$, we have:

$$
\begin{aligned}
l^2 &= 4W_i^2 - 4W_i a + a^2 \Rightarrow \\
E[R^2] &= (\frac{2+b}{3b} + \sqrt{\frac{8(1-p)}{3bp} + \Big(\frac{2+b}{3b}\Big)^2})(1/2 - 4/p) + \frac{8}{pb} + 1/p^2 \qquad (126)
\end{aligned}
$$

Finally the receiver variance is obtained by combining equations 121 and 126:

$$
Var(R) = E[(R - E[R])^2] = E[R^2] - E[R]^2 \qquad (127)
$$

APPENDIX B

MOBILITY PROTOCOL MODELS

In this section we define a simple analytical model that captures the latency induced by all the operational phases (registration, tunneling, and packet delivery) of mobility management protocols. Specifically, we develop analytical models for two promising mobile IP optimized protocol versions: Hierarchical Mobile IP (HMIP) and Mobile IP with route optimization (MIP-RO).

Packet Delivery: Packet delivery cost is crucial overhead in Mobile IP's performance, as packet tunneling is necessary even when the MH moves infrequently [103]. We calculate the overhead from packet delivery for MIP, HMIP, and MIP-RO.

Similar to [31, 114] we also define the following variables: s_h and s_g as the packet processing delays at the HA and GFA respectively. L_{ch}, L_{hg}, and L_{gf} are the latencies for delivering a packet from the CH to HA, HA to GFA, and GFA to FA respectively. Therefore, hierarchical mobile IP has a packet delivery overhead [114]:

$$L_{PD}^{hmip} = s_h + s_g + L_{hg} + L_{gf} + L_{ch} \tag{128}$$

The transmission delay from GFA to FA L_{gf}, is calculated as follows: $L_{gf} = l_{gf}\delta$, where δ is a proportionality constant and l_{gf} is the GFA/FA distance. If λ_a is the data packet arrival rate at the HA, then the packet processing delay is analogous to λ_a with $s_h = \nu\lambda_a$. In addition, the lookup overhead of the IP routing table has to be calculated which is analogous to its length e, the number of MHs in the subnet ω, and of course the packet arrival rate λ_a. Therefore this cost is $k\lambda_a log(k)$. We also assume that the distance between CH-HA and HA-GFA is the same, making thus $L_{hg} = L_{ch}$. So equation 128 becomes:

$$L_{PD}^{hmip} = \nu\lambda_a + e\lambda_a(a\omega k + log(k)) + (l_{gf} + 2 * l_{hg})\delta_D \tag{129}$$

On the other hand, MIP-RO does not suffer from triangular routing ($s_h = 0$) but it still has to suffer the tunneling overhead:

$$L_{PD}^{mipro} = s_g + L_{cg} + L_{gf} \tag{130}$$

Also s_g^{mipro} is the same with s_g^{hmip} because in addition to the routing overhead at the GFA, there is also the tunneling cost present (only s_h is avoided). With MIP-RO packets also have to be routed to the mobile host, and so the IP routing overhead does exist and that is why $s_g \neq 0$. Thus we have:

$$L_{PD}^{mipro} = \zeta k \lambda_a (a\omega k + log(k)) + (l_{gf} + l_{cg})\delta_D \tag{131}$$

Binding Updates: As mentioned earlier, binding or location updates are a requirement in Mobile IP even when the mobile host does not change its current address [83]. Let s_f, s_g, s_h, are the processing latencies of binding updates at the FA, GFA, and HA respectively, and LU_{hg}, LU_{gf}, LU_{fm}, are the transmission costs of binding updates between the HA-GFA, GFA-FA, and FA-MH respectively. Finally δ_U is a distance cost unit. So the latency due to a binding update to the HA and the local GFA are given by the following two equations:

$$LU_{uh}^{hmip} = 2\rho_f + 2\rho_g + \rho_h + 2LU_{hg} + 2LU_{gf} + 2LU_{fm} \tag{132}$$

$$LU_{ur}^{hmip} = 2s_f + s_g + 2l_{gf}\delta_U \tag{133}$$

For the MIP-RO case LU_{ch} will have to include the CH notification delay which is equal to $2l_{cg}$. So LU_{ch} is:

$$LU_{ch}^{mipro} = 2s_f + 2s_g + 2(l_{gf} + l_{cg})\delta_U \tag{134}$$

$$LU_{BU}^{mipro} = \frac{LU_{ch}}{T_f} \tag{135}$$

In order to calculate the expected disruption time for a specific scheme due to the latency caused by binding updates, we have to calculate the latency incurred by a specific network configuration. This means that for HMIP during handoff, the MH will only suffer the latency due to the the local binding update LU_{ur}^{hmip}. We can easily see that the disruption time becomes $t_h^{hmip} = LU_{ur}^{hmip}$. Similarly, for MIP-RO we have $t_h^{mipro} = LU_{ch}^{mipro}$.

REFERENCES

[1] "3GPP: Third Generation Partnership Project." http://www.3gpp.org.

[2] Ahmed, T., Mehaoua, A., Boutaba, R., and Iraqi, Y., "Adaptive Packet Video Streaming Over IP Networks: A Cross-Layer Approach," *IEEE Journal On Selected Areas In Communications*, vol. 23, no. 2, 2005.

[3] Akan, O. and Akyildiz, I., "Analytic Rate Control," *IEEE Transactions on Networking*, vol. 12, p. 634644, August 2004.

[4] Akyildiz, I. F. and others, "Mobility Management in Next-Generation Wireless Systems," *IEEE Proceedings*, vol. 87, pp. 1347–1384, August 1999.

[5] Altman, E., Avrachenkov, K., and Barakat, C., "A Stochastic Model of TCP/IP with Stationary Random Loss," in *SIGCOMM*, 2000.

[6] Altman, E., Avrachenkov, K., and Barakat, C., "TCP in presence of bursty losses," in *SIGMETRICS*, 2000.

[7] Apostolopoulos, J., Wong, T., Tan, W., and Wee, S., "On multiple description streaming with content delivery networks," in *INFOCOM*, 2002.

[8] Arauz, J. and Krishnamurthy, P., "Markov Modeling of 802.11 channels," in *IEEE Vehicular Technology Conference (VTC)*, 2003.

[9] Bakre, A. and Badrinath, B. R., "I-TCP: Indirect-TCP for Mobile Hosts," in *ICDCS*, 1995.

[10] Balakrishnan, H., *Challenges to Reliable Data Transport over Heterogeneous Wireless Networks*. PhD thesis, UC Berkeley, 1998.

[11] Barford, P. and Crovella, M., "Generating representative Web workloads for network and server performance evaluation," in *ACM SIGMETRICS*, 1998.

[12] Begen, A. C. and Altunbasak, Y., "Timely inference of late/lost packets in real-time streaming applications," in *Picture Coding Symposium (PCS)*, 2004.

[13] Begen, A. C. and Altunbasak, Y., "Videoconferencing over an intermediate-proxy," in *IEEE International Conference on Image Processing (ICIP)*, 2004.

[14] bender, P. and others, "CDMA/HDR: A Bandwidth Efficient High-Speed Wireless Data Service for Nomadic Users," *IEEE Communications Magazine*, pp. 70–77, July 2000.

[15] Blondia, C. and others, "Performance Analysis of Optimized Smooth Handoff in Mobile IP," in *MSWiM*, 2002.

[16] Blondia, C. and others, "Performance Comparison of Low Latency Mobile IP schemes," in *WiOpt*, 2003.

[17] Borsos, T., "Practical Model for VBR Video Traffic with Applications," in *IEEE International Conference on Management of Multimedia Networks and Services*, 2001.

[18] Boudec, J.-Y. L., *Performance Evaluation of Computer and Communication Systems*. http://icalwww.epfl.ch/perfeval/, 2005.

[19] Boukerche, A. and others, "A Two-Phase Handoff Management Scheme for Synchronizing Multimedia Units Over Wireless Networks," in *IEEE International Symposium on Computers and Communication (ISCC)*, 2003.

[20] Boukerche, A. and Harold Owens, I., "Media synchronization and QoS packet scheduling algorithms for wireless systems," *Mobile Networks and Applications*, vol. 10, no. 1-2, pp. 233–249, 2005.

[21] Cali, F., Conti, M., and Gregori, E., "IEEE 802.11 wireless LAN: capacity analysis and protocol enhancement," in *INFOCOM*, 1998.

[22] Cardwell, N. and others, "Modeling TCP Latency," in *INFOCOM*, 2000.

[23] Carpenter, B., "Architectural Principles of the Internet." RFC 1958, 1996.

[24] Cen, S. and others, "End-to-end differentiation of congestion and wireless losses," in *MMCN*, 2002.

[25] Chakareski, J. and Girod, B., "Rate-distortion Optimized Packet Scheduling and Routing for Media Streaming with Path Diversity," in *IEEE Data Compression Conference (DCC)*, 2003.

[26] Chakravorty, R., Cartwright, J., and Pratt, I., "Practical Experience with TCP over GPRS," in *GLOBECOM*, 2002.

[27] Chakravorty, R., Clark, A., , and Pratt, I., "Optimizing Web Delivery Over Wireless Links: Design, Implementation, and Experiences," *IEEE Journal On Selected Areas In Communications*, vol. 23, no. 2, 2005.

[28] Chan, M. C. and Ramjee, R., "TCP/IP Performance over 3G Wireless Links with Rate and Delay Variation," in *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 2002.

[29] Chen, J., Lv, T., and Zheng1, H., "Joint Cross-layer Design for Wireless QoS Content Delivery," in *ISCAS*, 2004.

[30] Chen, M. and Zakhor, A., "Transmission Protocols for Streaming Video over Wireless," in *IEEE International Conference on Image Processing (ICIP)*, 2004.

[31] Chen, W. and Ho, J., "Performance Analysis of Adaptive Location Management for Mobile IP," Tech. Rep. 97-CSE-13, Southern Methodist University, 1997.

[32] Chou, P. A. and Miao, Z., "Rate-distortion optimized streaming of packetized media," *IEEE Transactions On Circuits and Systems for Video Technology*, 2001.

[33] Chou, P. A. and Sehgal, A., "Rate-distortion optimized receiver-driven streaming over best-effort networks," in *IEEE International PacketVideo Workshop*, 2002.

[34] Crovella, M. and Bestavros, A., "Self-Similarity in World Wide Web Traffic: Evidence and Possible Cause," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 835–846, December 1996.

[35] Droms, R., "Dynamic Host Configuration Protocol." RFC 2131, March 1997.

[36] "Dummynet." http://info.iet.unipi.it/ luigi/ip_dummynet/.

[37] Etoh, M. and Yoshimura, T., "Advances in Wireless Video Delivery," *IEEE Proceedings*, vol. 93, January 2005.

[38] Floyd, S. and Fall, K., "Promoting the Use of End-to-End Congestion Control in the Internet," *IEEE/ACM Transactions on Networking*, August 1999.

[39] Frossard, P., "FEC Performance in Multimedia Streaming," *IEEE Communication Letters*, p. 122124, March 2001.

[40] Goyal, V. K., "Multiple Description Coding: Compression Meets the Network," *IEEE Signal Processing Magazine*, September 2001.

[41] Gupta, R., Chen, M., McCanne, S., and Walrand, J., "A Receiver-Driven Transport Protocol for the Web," in *Informs*, 2001.

[42] Gurtov, A., "Effect of Vertical Handovers on Performance of TCP-Friendly Rate Control," in *ACM Computer Communications Review*, 2004.

[43] Gurtov, A. and Floyd, S., "Modeling Wireless Links for Transport Protocols," *ACM Computer Communications Review*, vol. 34, pp. 85–96, April 2004.

[44] "H.263 codec." http://www.xs4all.nl/ roalt/h263.html.

[45] "ITU-T Recommendation H.264, Advanced video coding for generic audiovisual services," May 2003.

[46] Handley, M., Floyd, S., Pahdye, J., and Widmer, J., "TCP Friendly Rate Control (TFRC): Protocol Specification." RFC 3448, January 2003.

[47] Holland, G. and Vaidya, N., "Analysis of TCP performance over mobile ad hoc networks," in *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 1999.

[48] Holma, H. and Toskala, A., *WCDMA for UMTS*. Wiley, 2nd ed., 2002.

[49] Hsiao, P.-H., Kung, H., and Tan, K.-S., "Video over TCP with Receiver-based Delay Control," in *ACM NOSSDAV*, 2001.

[50] "IEEE Standard for Local and Metropolitan Area Networks Part 16: Air Interface." IEEE Std 802.16, 2001.

[51] Izquierdo, M. R. and Reeves, D. S., "A survey of statistical source models for variable-bit-rate compressed video," *Multimedia Systems*, 1999.

[52] Jacobson, V., "Congestion Avoidance and Control," in *ACM SIGCOMM*, pp. 314–329, August 1988.

[53] Kalman, M., Ramanathan, P., and Girod, B., "Rate-distortion optimized video streaming with multiple deadlines," in *IEEE International Conference on Image Processing (ICIP)*, vol. 3, pp. 661–664, September 2003.

[54] Kalman, M., Steinbach, E., and Girod, B., "Adaptive playout for real-time media streaming," in *IEEE International Conference on Image Processing (ICIP)*, October 2001.

[55] Kalman, M. and others, "Adaptive Media Playout for Low-Delay Video Streaming Over Error-Prone Channels," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, p. 841851, June 2004.

[56] Karrer, R. and Gross, T., "Dynamic Handoff of Multimedia Streams," in *NOSSDAV*, 2001.

[57] Karrer, R. and Gross, T., "Multipath streaming in best-effort networks," in *IEEE International Conference on Communications (ICC)*, 2003.

[58] Kim, M. Y., "Rate-Distortion Comparisons between FEC and MDC based on Gilbert Channel Model," in *International Conference on Networks (ICON)*, 2003.

[59] Kolding and others, "High-speed Downlink Packet Access: WCDMA Evolution," *IEEE Vehicular Society Technolgy News*, vol. 50, no. 1, pp. 4–10, 2003.

[60] Krasic, C., Li, K., and Walpole, J., "The Case for Streaming Multimedia with TCP," in *Workshop on Interactive Distributed Multimedia Sytems (IDMS)*, (Lancaster, U.K.), September 2001.

[61] Krunz, M. and Tripathi, S. K., "On the characterization of VBR MPEG streams," in *ACM SIGMETRICS*, 1997.

[62] Lakshman, T. and Madhow, U., "The performance of TCP/IP for networks with high bandwidth-delay products," *IEEE/ACM Transactions on Networking*, vol. 5, pp. 336–350, june 1997.

[63] Larzon, L.-A. and others, "The Lightweight User Datagram Protocol (UDP-Lite)." RFC 3828, July 2004.

[64] Lee, Y.-C., *Error Resilient Video Streaming over Lossy Networks*. PhD thesis, Georgia Institute of Technology, 2003.

[65] Lee, Y.-C., Kim, J., Altunbasak, Y., and Mersereau, R., "Performance Comparisons of Layered and Multiple Description Coded Video Streaming over Error-Prone Networks," in *ICC*, 2003.

[66] Li, B., Li, L., Li, B., and Cao, X.-R., "On handoff performance for an integrated voice/data cellular system," *Kluwer Wireless Networks*, vol. 9, no. 4, pp. 393–402, 2003.

[67] Li, J.-S., "Modeling VBR Traffic with Autoregressive Gaussian Processes," in *IEEE International Conference on Networks (ICON)*, 2000.

[68] Liang, S. and Cheriton, D., "TCP-RTM: Using TCP for Real Time Applications," in *ICNP*, 2002.

[69] Liang, Y. and Girod, B., "Prescient RD optimized packet dependency management for low-latency video streaming," in *IEEE International Conference on Image Processing (ICIP)*, vol. 2, pp. 659–662, September 2003.

[70] Manzoni, P., Cremonesi, P., and Serazzi, G., "Workload Models of VBR Video Traffic and Their Use in Resource Allocation Policies," *IEEE/ACM Transactions On Networking*, vol. 7, pp. 387–397, June 1999.

[71] McNair, J., Akyildiz, I., and Bender, M., "Handoffs for Real-Time Traffic in Mobile IP Version 6 Networks," in *IEEE ICC*, 2001.

[72] Mehra, P. and Zakhor, A., "TCP-Based Video Streaming Using Receiver-Driven Bandwidth Sharing," in *International Packet Video Workshop*, 2003.

[73] Miao, Z. and Ortega, A., "Optimal Scheduling for Streaming of Scalable Media," in *Asilomar Conference on Signals, Systems, and Computers*, 2001.

[74] Misra, V. and others, "Stochastic Differential Equation Modeling and Analysis of TCP-Windowsize Behavior," tech. rep., University of Massachussets: ECE-TR-CCS-99-10-01, 1999.

[75] Monks, J., Sinha, P., and Bharghavan, V., "Limitations of TCP-ELFN for ad hoc networks," in *MOMUC*, October 2000.

[76] Mukherjee, B. and Brecht, T., "Time-Lined TCP for the TCP-Friendly Delivery of Streaming Media," in *ICNP*, 2000.

[77] "Network simulator."

[78] Originization, I. S. http://www.iso.org.

[79] Pack, S. and Choi, Y., "Performance Analysis of Hierarchical Mobile IPv6 in IP-based Cellular Networks," in *PIMRC*, 2003.

[80] Padhye, J., Firoiu, V., and Towsley, D., "Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation," *IEEE/ACM Transactions on Networking*, vol. 8, pp. 133–145, April 2000.

[81] Parikh, H. and others, "Seamless Handoff of Mobile Terminal from WLAN to cdma2000 Network," 2003.

[82] Paxson, V., "Empirically Derived Analytical Models of Wide-Area TCP Connections," *IEEE/ACM Transactions on Networking*, vol. 2, August 1994.

[83] Perkins, C., "IP Mobility Support for IPv4." RFC 3220, January 2002.

[84] Postel, J., "Transmission Control Protocol." RFC 793, 1981.

[85] "Quicktime." http://www.apple.com.

[86] Rappaport, T., Seidel, S., and Takamizawa, K., "Statistical Channel Impulse Response Models for Factory and Open Plan Building Radio Communication System Design," *IEEE Transactions on Communications*, vol. 39, p. 794807, May 1991.

[87] "Real media." http://www.realnetworks.com.

[88] Reinbold, P. and Bonaventure, O., "A Survey of IP micro-mobility protocols." citeseer.nj.nec.com/reinbold02survey.html, 2002.

[89] Ross, S., *Introduction to probability models.* Academic Press, 2003.

[90] S. Low, L. P. and Wang, L., "Understanding TCP Vegas: A duality model," in *SIGMETRICS*, 2001.

[91] Samios, C. B. and Vernon, M. K., "Modeling the throughput of TCP Vegas," in *International conference on Measurement and modeling of computer systems (SIGMETRICS)*, 2003.

[92] Schulzrinne, Casner, Frederick, and Jacobson, "RTP: A Transport Protocol for Real-Time Applications." draft-ietf-avt-rtp-new-12.txt, March 2003.

[93] Schulzrinne, H. and others, "Real Time Streaming Protocol (RTSP)." RFC 2326, 1998.

[94] "TML Video sequences." Available from http://kbs.cs.tu-berlin.de/ stewe/vceg/sequences.htm.

[95] Sikdar, B., Kalyanaraman, S., and Vastola, K., "Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno and SACK," in *INFOCOM*, 2001.

[96] Sinha, P., Venkitaraman, N., Sivakumar, R., and Bharghavan, V., "WTCP: A reliable Transport Protocol for Wireless Wide-Area Networks," in *ACM International Conference on Mobile Computing and Networking (MOBICOM)*, 1999.

[97] Soong, A. and others, "Forward High-Speed Wireless Packet Data Service in IS-2000 – 1xEV-DV," *IEEE Communications Magazine*, August 2003.

[98] Steinbach, E., Liang, Y., and Girod, B., "A simulation study of packet path diversity for TCP file transfer and media transport on the Internet," in *Tyrrhenian International Workshop on Digital Communications (IWDC)*, 2002.

[99] Stevens, R., *TCP/IP Illustrated Volume 1.* Addison-Wesley, 1994.

[100] Stewart, R., Ramalho, M., Xie, Q., Tuexen, M., and Conrad, P., "SCTP Partial Reliability Extension." draft-ietf-tsvwg-prsctp-00.txt, June 2003.

[101] Stewart, R., Xie, Q., Morneault, K., Sharp, C., Schwarzbauer, H., Taylor, T., Rytina, I., Kalla, M., Zhang, L., and Paxson, V., "Stream Control Transmission Protocol." RFC 2960, October 2000.

[102] Stockhammer, T., Jenkac, H., and Kuhn, G., "Streaming Video Over Variable Bit-Rate Wireless Channels," *IEEE Transactions on Multimedia, Special Issue on Streaming Media*, vol. 6, pp. 268–277, April 2004.

[103] Stojmenovic, I., *Handbook of Wireless Networks and Mobile Computing.* John Wiley and Sons, February 2002.

[104] Sun, M.-T. and Reibman, A. R., *Compressed Video Over Networks*. Marcel Dekker Inc., September 2001.

[105] Tan, D. and Zahkor, A., "Real-time Internet Video Using Error Resilient Scalable Compression and TCP-friendly ransport Protocol," *IEEE Trans. on Multimedia*, May 1999.

[106] Tian, D., Lee, Y.-C., AlRegib, G., and Altunbasak, Y., "Packetized Media Streaming over Multiple Wireless Channels," in *ICC*, 2004.

[107] Tsukamoto, K., Hori, Y., and Oie, Y., "Mobility management of transport protocol supporting multiple connections," in *The second international workshop on mobility management & wireless access protocols (Mobiwac)*, 2004.

[108] Venkataram, P., Rajavelsamy, R., and Laxmaiah, S., "A method of data transfer control during handoffs in mobile-IP based multimedia networks," *ACM Computer Communications Review*, vol. 5, no. 2, pp. 27–36, 2001.

[109] Wang, T. C. and others, "Low-Delay and Error-Robust Wireless Video Transmission for Video Communications," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, pp. 1049–1058, December 2002.

[110] Wang, Y., Ostermann, J., and Zhang, Y.-Q., *Video Processing and Communications*. Prentice Hall, 2002.

[111] "Windows media." http://www.microsoft.com.

[112] Wu, D., Hou, T., Zhu, W., Lee, H.-J., Chiang, T., Zhang, Y.-Q., and Chao, H. J., "On End-to-End Architecture for Transporting MPEG-4 Video over the Internet," *IEEE Trans. on Circuits and Systems for Video Technology*, September 2000.

[113] Wu, D. and others, "An End-to-End Approach for Optimal Mode Selection in Internet Video Communication: Theory and Application," *IEEE JSAC*, vol. 18, pp. 977–995, June 2000.

[114] Xie, J. and Akyildiz, I. F., "A Novel Distributed Location Management Scheme for Minimizing Signaling Costs in Mobile IP," *IEEE Transactions on Mobile Computing*, vol. 1, pp. 163–175, July-September 2002.

[115] Yang, Y., Kim, M., and Lam, S., "Transient Behaviors of TCP-friendly Congestion Control Protocols," in *INFOCOM*, 2001.

[116] Z., M. and Ortega, A., "Expected Runtime Distortion Based Scheduling for Delivery of Scalable Media," in *International Conference of Packet Video*, 2002.

[117] Zhang, Q., Zhu, W., and Zhang, Y., "Resource Allocation for Multimedia Streaming over the Internet," *IEEE Transactions on Multimedia*, vol. 3, pp. 339–335, September 2001.

[118] Zhang, Y., Paxson, V., and Shenker, S., "The stationarity of internet path properties: Routing, loss, and throughput.," tech. rep., ACIRI, 2000.

[119] Zorzi, M., Rao, R. R., and Milstein, L. B., "ARQ error control for fading mobile radio channels," *IEEE Transactions on Vehicular Technology*, vol. 46, pp. 445–455, May 1997.